KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, 3001 Leuven (Heverlee)

REALIZATION, IDENTIFICATION AND FILTERING FOR HIDDEN MARKOV MODELS USING MATRIX FACTORIZATION TECHNIQUES

Promotors:
Prof.dr.ir. B. De Moor
Prof.dr.ir. J.C. Willems

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Bart VANLUYTEN

Mei 2008

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, 3001 Leuven (Heverlee)

REALIZATION, IDENTIFICATION AND FILTERING FOR HIDDEN MARKOV MODELS USING MATRIX FACTORIZATION TECHNIQUES

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Bart VANLUYTEN

U.D.C. 681.5.015 Mei 2008

ISBN 978-90-5682-932-2

U.D.C. 681.5.015

D/2008/7515/41

Voorwoord

Aan het eind van mijn doctoraat wil ik een aantal mensen bedanken zonder wie dit werk nooit tot stand zou zijn gekomen.

In de eerste plaats bedank ik professor Bart De Moor om als promotor voor deze thesis te willen optreden. Dankzij zijn enthousiasme en verfrissende ideeën slaagde hij erin om een grote en succesvolle onderzoeksgroep uit te bouwen met specialisaties in verschillende domeinen, waarin hij ook voor mij een plaats heeft willen voorzien. Bij het begin van mijn doctoraatsonderzoek heb ik de mogelijkheid gehad om samen met Bart een literatuuroverzicht over realisatietheorie op te stellen. Eén onderdeel van dat overzicht, realisatie voor hidden Markov modellen, is het startpunt van mijn onderzoek geworden.

Na ongeveer een jaar onderzoek leerde ik professor Jan Willems kennen. Sinds een paar jaar emeritus, maar actiever dan ooit, was hij direct geïnteresseerd om mijn doctoraatsproject mee te begeleiden en uiteindelijk copromotor te worden. Gedurende ongeveer 4 jaar zag ik Jan ongeveer wekelijks om te discussiëren over mijn onderzoek. Zijn scherpe intuïtie, erg ruime interesse en kritische ingesteldheid hebben in belangrijke mate bijgedragen aan dit doctoraat.

Ik bedank de professoren A. Bultheel, P. Spreij en K. Meerbergen om zich als juryleden te willen engageren. Zij gaven me waardevolle opmerkingen en suggesties die mijn doctoraatstekst aanzienlijk verbeterd hebben. *I would like to thank professors V. Blondel and L. Finesso to be members of my examination jury. They both provided me with valuable comments and suggestions which improved my PhD text.* Professor H. Van Brussel wil ik bedanken om voorzitter van mijn doctoraatsjury te willen zijn.

Het Fonds voor Wetenschappelijk Onderzoek (FWO) ben ik erkentelijk voor het ter beschikking stellen van een doctoraatsbeurs, die me toeliet om zorgeloos aan mijn onderzoek te werken.

Om een doctoraat aan de KULeuven te behalen moet een ganse administratieve lijdensweg worden afgelegd. Gelukkig waren er Ida, Péla en Ilse om me feilloos langs de administratieve valkuilen te loodsen, waarvoor ik hen ten zeerste dankbaar ben.

Verder ben ik de vele SCD-collega's dankbaar voor de erg fijne werksfeer binnen de groep. Een aantal van hen zijn gedurende mijn doctoraatsperiode dan ook echte vrienden geworden, ik denk hierbij aan mijn eilandgenootjes Jeroen, Steven en Erik en onze schuin-onder-buurman Tom. Ik denk ook met plezier

terug aan de jaarlijkse SISTA-weekends die me er telkens weer aan herinnerden wat voor leuke, hechte groep we vormen.

Ik bedank mijn toffe vriendengroep van Tienen voor de vele ontspannende momenten tijdens de weekends. Deden we nu een spellekesavond, gingen we samen eten, was er een house-warming of werd er een verjaardag gevierd, altijd amuseerden we ons even goed. Tijdens de laatste periode van mijn doctoraat heb ik echter heel wat afspraken moeten afzeggen omwille van de drukte typisch aan de eindfase van een doctoraat. Zoals het echte vrienden siert, hebben jullie hiervoor alle begrip getoond. Bedankt daarvoor! Ik bedank ook Luc voor zijn voortdurende interesse in de stand van zaken van mijn onderzoek.

Ik dank mijn bomma, nonkels, tantes, Sofie en Hans om er altijd voor me geweest te zijn. Mijn grote broer Jan ben ik dankbaar voor onze babbels tijdens de weekends waarin we onze belevenissen van de voorbije week uitwisselden. Ik bedank ook Jeanne en Guy voor hun interesse in m'n doctoraatswerk.

Mijn ouders wil ik bedanken voor de vele kansen die ze me gegeven hebben en die uiteindelijk geleid hebben tot het behalen van dit doctoraat. Papa, bedankt voor de leuke momenten die we vroeger samen beleefd hebben. Ik weet dat je vandaag ongetwijfeld fier op me zou geweest zijn. Mama, bedankt voor je onvoorwaardelijk hulp gedurende de ganse periode van mijn doctoraat. Je aanmoedigingen en optimisme zijn een enorme steun voor me geweest. Bedankt!

En tenslotte, Stijneke, naar jou gaat een speciaal woordje van dank voor je lieve zorgen van elke dag en je aanmoedigingen om door te zetten. Laat ons vanaf vandaag verder bouwen aan een mooie toekomst voor ons twee...

Bart Vanluyten,
Leuven, mei 2008.

Abstract

Since their introduction in 1957, hidden Markov models have been used in several engineering applications (speech processing, computational biology). However, many theoretical questions concerning hidden Markov models remain open until this moment. Contributing to these theoretical questions forms the first main objective of this thesis. When considering the theoretical problems, we find inspiration in the analogy with the corresponding problems for linear stochastic models. The solution to most of the problems concerning linear stochastic models makes use of the singular value decomposition. For the solution of the corresponding problems for hidden Markov models, it turns out that modifications to the nonnegative matrix factorization are needed. Investigating new nonnegative matrix factorization techniques forms the second main objective of this thesis.

A first theoretical problem concerning hidden Markov models is the exact positive realization problem. No procedure is known to solve this problem. In this thesis, two relaxed versions of the problem are solved: the exact quasi realization problem and the approximate positive realization problem. A second problem is the identification problem for hidden Markov models. In this thesis we propose an identification approach that estimates the state sequence directly from the output data and subsequently computes the system matrices from the obtained state sequence and the given output sequence. This approach is analogous to subspace identification for linear stochastic models. A third problem is the estimation problem for hidden Markov models. We show that it suffices for several types of estimation problems to have a solution to the quasi realization problem instead of a solution to the positive realization problem. The techniques are applied to the detection of motifs in DNA sequences.

Concerning the second objective, we consider two modifications to the nonnegative matrix factorization: the structured nonnegative matrix factorization and the nonnegative matrix factorization without nonnegativity constraints on the factors. It turns out that these factorizations are applicable in engineering applications, apart from the hidden Markov research. The structured nonnegative matrix factorization is applied to the clustering of data points based on their distance matrix. The nonnegative matrix factorization problem without nonnegativity constraints on the factors is applied to the modeling of a database containing human faces.

Korte inhoud

Sinds hun introductie in 1957 worden verborgen Markov modellen veelvuldig gebruikt in ingenieurstoepassingen (spraakherkenning, biologie). Ondanks de vele toepassingen blijven tot nu toe nog een heel aantal theoretische vragen omtrent verborgen Markov modellen open. Bijdragen aan deze theoretische problemen vormt de eerste doelstelling van deze thesis. Bij het oplossen van problemen omtrent verborgen Markov modellen kan inspiratie gezocht worden in de oplossing van de overeenkomstige problemen voor lineair stochastische modellen. De oplossing van de meeste problemen betreffende lineair stochastische modellen maakt gebruik van de singuliere-waardenontbinding. Voor de problemen aangaande verborgen Markov modellen blijken varianten op de niet-negatieve matrixontbinding nodig. Het onderzoek naar nieuwe niet-negatieve matrixontbindingen is de tweede doelstelling van dit proefschrift.

Een eerste theoretisch probleem aangaande verborgen Markov modellen is het exacte positieve realisatieprobleem. Er is geen procedure gekend om dit probleem op te lossen. In deze thesis worden twee afgezwakte versies van dit probleem opgelost: het exacte quasi-realisatieprobleem en het benaderende positieve realisatieprobleem. Een tweede probleem is het identificatieprobleem voor verborgen Markov modellen. In deze thesis stellen we een identificatiemethode voor die de toestandssequentie rechtstreeks uit de uitgangsdata schat en vervolgens de modelparameters berekent uit de bekomen toestandssequentie en de gegeven uitgangssequentie. Deze aanpak is analoog aan deelruimte-identificatie voor lineair stochastische modellen. Een derde probleem is het schattingsprobleem voor verborgen Markov modellen. We tonen aan dat het voor verschillende types van schattingsproblemen volstaat om een oplossing te hebben voor het quasi-realisatieprobleem in plaats van een oplossing voor het positieve realisatieprobleem. We passen de methodes toe op het detecteren van motieven in DNA-sequenties.

Betreffende de tweede doelstelling, stellen we twee varianten op de niet-negatieve matrix ontbinding voor: de gestructureerde niet-negatieve matrixontbinding en de niet-negatieve ontbinding zonder niet-negativiteitsbeperkingen op de factoren. Beide ontbindingen hebben nut op zich, los van het onderzoek naar verborgen Markov modellen. We passen de gestructureerde niet-negatieve matrixontbinding toe op het clusteren van datapunten. De ontbinding zonder niet-negativiteitsbeperkingen op de factoren wordt gebruikt voor het modelleren van menselijke aangezichten.

Glossary

Variables

a, b, c	Vector variables
A, B, C	Matrix variables
$\mathbf{1}_{m,n}$	Matrix of size $(m \times n)$ with all elements equal to 1
e_m	Column vector with all elements equal to 1, if no confusion is possible, we use e instead of e_m
I_m	Identity matrix of size $(m \times m)$

Sets

$\mathbb{A}, \mathbb{B}, \mathbb{C}$	Sets
$\lvert\mathbb{A}\rvert$	Cardinality of the set $\mathbb{A}$
$\mathbb{N}$	Set of natural numbers $\{1, 2, \ldots\}$
$\mathbb{Z}$, $\mathbb{Z}_+$	Set of integers, nonnegative integers $\{0, 1, \ldots\}$
$\mathbb{R}$, $\mathbb{R}_+$	Set of real numbers, nonnegative real numbers
$\mathbb{R}^n$, $\mathbb{R}^n_+$	Set of n-dimensional vectors with elements from $\mathbb{R}$, $\mathbb{R}_+$
$\mathbb{R}^{m \times n}$, $\mathbb{R}^{m \times n}_+$	Set of $m \times n$ matrices with elements from $\mathbb{R}$, $\mathbb{R}_+$

Matrix operations

$A^\top$	Transpose of matrix A
A^{-1}	Inverse of matrix A
$A^\dagger$	Moore-Penrose pseudo-inverse of A
$A_{i,j}$	Element at the i-th row and j-th column of A
$A_{i,:}$	i-th row of A
$A_{:,j}$	j-th column of A
$A_{i:j,k:l}$	Submatrix of A bounded by the i-th and j-th row and by the k-th and l-th column of A
$A \geq 0$	Matrix A is elementwise nonnegative
$A \succeq 0$	Matrix A is nonnegative definite
$A \otimes B$	Kronecker product of A and B
$\mathrm{diag}(a_1, a_2, \ldots)$	Diagonal matrix with $a_1, a_2, \ldots$ on its diagonal
$\mathrm{diag}(a)$	Diagonal matrix with the vector a on its diagonal
$\mathrm{vec}(A)$	Row vector whose elements are row-wise scanned from A
$*$	Element, subvector or submatrix of a matrix of which the exact value is unimportant

Norms of and distances between matrices

$\|\|X\|\|$	Norm of X
$D(X, Y)$	Distance between X and Y
$\|\|X\|\|_F$	Frobenius norm of X
$\|\|X - Y\|\|_F$	Frobenius distance between X and Y
$D_{\mathrm{KL}}(X\|\|Y)$	Kullback-Leibler divergence between matrix X and Y

Optimization

$\min_{\dot{x}}$	Function minimization over $\dot{x}$, optimal function value is returned
$\mathrm{argmin}_{\dot{x}}$	Function minimization over $\dot{x}$ optimal value of $\dot{x}$ is returned
s.t.	Subject to the constraints

Probability, expected value

$P(x)$	Probability that the event x occurs
$E(x)$	Expected value of the random vector x
$\mathcal{E}(x)$	Estimate of the random vector x
$ML(x)$	Most likely estimate of the random vector x

Miscelaneous

$\delta(k,l)$	Kronecker delta: $\begin{cases} \delta(k,l)=1 & \text{if } k=l \\ \delta(k,l)=0 & \text{if } k\neq l \end{cases}$

Finite valued strings

$\mathsf{a}, \mathsf{b}, \mathsf{c}$	Symbol from finite set $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$
$\mathbf{a}, \mathbf{b}, \mathbf{c}$	String taking values in a finite set
$\mathcal{A}, \mathcal{B}, \mathcal{C}$	Ordered sets of strings
a_i	i-th symbol of string $\mathbf{a}$
$\lvert\mathbf{a}\rvert$	Length of string $\mathbf{a}$
$\mathbf{a}^{(1)}\mathbf{a}^{(2)}$	Concatenation of string $\mathbf{a}^{(1)}$ and string $\mathbf{a}^{(2)}$

Fixed symbols

$\mathcal{P}(\mathbf{x})$	Probability of the string $\mathbf{x}$
$\pi(1)$	Initial state distribution of a HMM
$\Pi_{\mathbb{X}}$	State transition matrix of a HMM
Π	Output and next state mapping of a Mealy HMM
β	Output mapping of a Moore HMM

Acronyms

HMM	Hidden Markov Model
LSM	Linear Stochastic Model
SVD	Singular Value Decomposition
NMF	Nonnegative Matrix Factorization

Contents

Nederlandse samenvatting

Realisatie, identificatie en filtering voor verborgen Markov modellen gebruik makende van matrixontbindingstechnieken

Hoofdstuk 1: Inleiding

Een systeem is een fysisch, economisch, biologisch, industrieel, technisch, ... fenomeen dat interageert met zijn omgeving. Het gedrag van systemen wordt gewoonlijk onderzocht gebruik makende van wiskundige modellen. Een wiskundig model beschrijft de relatie tussen verschillende in- en uitgangen van het systeem als functie van de tijd. Een eerste grote doelstelling van dit proefschrift is het bestuderen van een specifieke modelklasse: *verborgen Markov modellen* (HMM). Hoewel deze modellen erg veel worden toegepast voor het bestuderen van allerlei ingenieursproblemen, blijven een heel aantal theoretische vragen aangaande verborgen Markov modellen onopgelost tot op dit ogenblik. In Hoofdstuk 3 worden verborgen Markov modellen op een formele manier gedefinieerd. Vervolgens worden een aantal theoretische problemen over verborgen Markov modellen beschouwd: het quasi-realisatieprobleem voor HMMs (Hoofdstuk 4), het positieve realisatieprobleem voor HMMs (Hoofdstuk 5), het identificatieprobleem voor HMMs (Hoofdstuk 6) en het schattingsprobleem voor HMMs (Hoofdstuk 7).

Verborgen Markov modellen zijn nauw verwant aan *lineair stochastische modellen* waarvoor het theoretisch onderzoek een zekere graad van maturiteit heeft bereikt. In Hoofdstuk 3 worden lineair stochastische modellen op een formele manier gedefinieerd. Bij het oplossen van problemen aangaande verborgen Markov modellen maken we voortdurend gebruik van de kennis van de oplossing van het overeenkomstige probleem voor lineair stochastische modellen.

De oplossing van de meeste theoretische problemen betreffende lineair stochastische modellen maakt gebruik van de singuliere-waardenontbinding, een populaire matrixontbindingstechniek. Het zal blijken dat er voor het oplossen van de theoretische problemen aangaande verborgen Markov modellen, nood is aan de niet-negatieve matrixontbinding en varianten op deze ontbinding. Het ontwikkelen van varianten op de niet-negatieve matrixontbinding vormt dan ook de tweede grote doelstelling van dit proefschrift. We bespreken matrixontbindingtechnieken in Hoofdstuk 2.

Hoofdstuk 2: Matrixontbindingen

In dit hoofdstuk worden eerst twee bestaande matrixontbindingsmethodes besproken: de singuliere-waardenontbinding en de niet-negatieve matrixontbinding. Daarna worden twee aanpassingen aan de niet-negatieve matrixontbinding voorgesteld, namelijk de gestructureerde niet-negatieve matrixontbinding en de niet-negatieve matrixontbinding zonder niet-negativiteitsbeperkingen op de factoren.

Singuliere-waardenontbinding

De *singuliere-waardenontbinding (SVD)* van een gegeven matrix $M \in \mathbb{R}^{m_1 \times m_2}$ van rang r wordt gegeven door

$$M = U \Sigma V^\top$$

waarbij $U \in \mathbb{R}^{m_1 \times m_1}$ en $V \in \mathbb{R}^{m_2 \times m_2}$ unitaire matrices zijn en

$$\Sigma = \begin{bmatrix} \Sigma_{(1)} & 0 \\ 0 & 0 \end{bmatrix}$$

met

$$\Sigma_{(1)} := \operatorname{diag}(\sigma_1(M), \sigma_2(M), \ldots, \sigma_r(M)),$$

met

$$\sigma_1(M) \geq \sigma_2(M) \geq \ldots \geq \sigma_r(M) > 0.$$

De rang-k SVD-benadering van M (met $k \leq r$) is gedefiniëerd als

$$M_k := U \begin{bmatrix} \Sigma_k & 0 \\ 0 & 0 \end{bmatrix} V^\top$$

waarbij $\Sigma_k := \operatorname{diag}(\sigma_1(M), \sigma_2(M), \ldots, \sigma_k(M))$. Men kan aantonen dat de rang-k SVD-benadering van M aanleiding geeft tot een optimale rang-k benadering van M in de Frobenius afstand. Bovendien is het zo dat als de zogenaamde *gapconditie* $\sigma_k(M) > \sigma_{k+1}(M)$ voldaan is, dat de rang-k SVD-benadering M_k de unieke matix van rank k is die M optimaal benadert in de Frobenius afstand. We hebben aangetoond dat, indien de gapconditie voldaan is, de rang-k benadering van een matrix die voldoet aan de symmetrie $M = PMQ$, voldoet aan dezelfde symmetrie, i.e. $M_k = PM_kQ$.

Niet-negatieve matrixontbinding

Het niet-negatieve matrixontbindingsprobleem is gedefinieerd als volgt: gegeven een matrix $M \in \mathbb{R}_+^{m_1 \times m_2}$, vind een ontbinding $M = VH$ waarbij $V \in \mathbb{R}_+^{m_1 \times a}$ en $H \in \mathbb{R}_+^{a \times m_2}$, en met a zo klein mogelijk. De minimale innerdimensie van een exacte positieve matrixontbinding wordt de positieve rang genoemd. Er bestaat een eindig algoritme voor het berekenen van de positieve rang. Dit tijdscomplexiteit van dit algoritme is echter niet-polynomiaal. Lee en Sueng introduceerden daarom de benaderende niet-negatieve matrixontbinding [72]. Het idee bestaat erin dat men de inwendige dimensie a kiest en vervolgens niet-negatieve matrices V en H zoekt, zodaning dat VH een optimale benadering is voor M volgens een zeker criterium. The Kullback-Leibler divergentie is een populaire afstandsmaat tussen niet-negatieve matrices en is gedefineerd als

$$D_{KL}(A||B) := \sum_{ij}(A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij}).$$

De benaderende niet-negatieve matixontbinding kan als volgt worden gedefinieerd

Problem 0.1. *Gegeven $M \in \mathbb{R}_+^{m_1 \times m_2}$ en gegeven a, minimaliseer $D_{KL}(M||VH)$ met betrekking tot V (van grootte $m_1 \times a$) en H (van grootte $a \times m_2$), zodaning dat $V \geq 0$, $H \geq 0$.*

Dit probleem is niet-convex in V en H samen en wordt daarom typisch opgelost door alternerende iteratieve methodes waar eerst een update van V wordt doorgevoerd, vervolgens een update van H, dan weer V enzovoort. In [72, 73] worden iteratieve formules gegeven om Probleem 0.1 op te lossen.

Gestructureerde niet-negatieve matrixontbinding

De benaderende gestructureerde niet-negatieve matrixontbinding wordt gedefinieerd als

Problem 0.2. *Gegeven $P \in \mathbb{R}_+^{p \times p}$ en gegeven a, minimaliseer $D_{KL}(P||VAV^\top)$ met betrekking tot V (van grootte $p \times a$) en A (van grootte $a \times a$), zodanig dat $V \geq 0$, $H \geq 0$.*

In dit proefschrift wordt aangetoond dat een stationair punt (A, V) van de kostfunctie $D_{KL}(P||VAV^\top)$ het gemiddelde van de rij- en kolomsom van P behoudt, i.e.

$$\frac{\sum_l P_{kl} + P_{lk}}{2} = \frac{\sum_l (VAV^\top)_{kl} + (VAV^\top)_{lk}}{2}, \quad k = 1, 2, \ldots p.$$

Als gevolg daarvan wordt de elementsom van P ook bewaard, i.e.

$$\sum_{kl} P_{kl} = \sum_{kl} (VAV^\top)_{kl}.$$

Vervolgens stellen we iteratieve formules voor en tonen we aan dat de divergentie $D_{KL}(P||VAV^\top)$ niet-stijgend is onder deze formules.

$$A_{ij}^{(t+1)} = A_{ij}^{(t)} \sum_{\mu\nu} V_{\mu i} V_{\nu j} \frac{P_{\mu\nu}}{(VA^{(t)}V^\top)_{\mu\nu}}, \tag{0.1}$$

$$V_{ki}^{(t+1)} = V_{ki}^{(t)} \frac{\sum_{\lambda\nu} \frac{P_{k\nu}}{(V^{(t)}A(V^{(t)})^\top)_{k\nu}} A_{i\lambda} V_{\nu\lambda}^{(t)} + \frac{P_{\nu k}}{(V^{(t)}A(V^{(t)})^\top)_{\nu k}} A_{\lambda i} V_{\nu\lambda}^{(t)}}{\sum_{\lambda\mu\nu} \frac{P_{\mu\nu}}{(V^{(t)}A(V^{(t)})^\top)_{\mu\nu}} A_{i\lambda} V_{\nu\lambda}^{(t)} V_{\mu i}^{(t)} + \frac{P_{\nu\mu}}{(V^{(t)}A(V^{(t)})^\top)_{\nu\mu}} A_{\lambda i} V_{\nu\lambda}^{(t)} V_{\mu i}^{(t)}}. \tag{0.2}$$

Tenslotte tonen we aan dat de divergentie invariant is onder de aanpassingen (0.1) en (0.2) als en slechts als (A, V) een stationair punt is van de divergentie, i.e.

$$\left\{ \begin{array}{l} A^{(t+1)} = A^{(t)}, \\ V^{(t+1)} = V^{(t)}, \end{array} \right. \Leftrightarrow \left\{ \begin{array}{ll} A_{ij}^{(t)} \frac{\partial F}{\partial A_{ij}}(A^{(t)}, V^{(t)}) = 0, & i = 1, 2, \dots a; j = 1, 2, \dots a, \\ V_{ki}^{(t)} \frac{\partial F}{\partial V_{ki}}(A^{(t)}, V^{(t)}) = 0, & k = 1, 2, \dots p; i = 1, 2, \dots a. \end{array} \right.$$

We passen de gestructureerde niet-negatieve matrixontbinding toe op het clustering probleem. In dat probleem zijn een aantal datapunten gegeven en is het de bedoeling om *clusters* van punten te zoeken die dicht bij elkaar liggen. Als data voor het probleem hebben we de afstandsmatrix tussen de verschillende punten. Door de gestructureerde niet-negatieve matrixontbinding toe te passen op deze afstandsmatrix, wordt een opdeling van de datapunten in a clusters bekomen.

Niet-negatieve matrixontbinding zonder niet-negativiteitsbeperkingen op de factoren

De niet-negatieve matrixontbinding zonder niet-negativiteitsbeperkingen op de factoren wordt gedefinieerd als

Problem 0.3. *Gegeven $M \in \mathbb{R}_+^{m_1 \times m_2}$ en $a \in \mathbb{N}$. Minimaliseer $D_{\mathrm{KL}}(M||VH)$ met betrekking tot V $(\in \mathbb{R}^{m_1 \times a})$ en H $(\in \mathbb{R}^{a \times m_2})$, zodanig dat $VH \geq 0$.*

Merk op dat er geen niet-negativiteitsbeperkingen zijn op de matrices V en H zelf, maar enkel op het product VH. Het is intuïtief duidelijk dat de niet-negatieve matrixontbinding zonder niet-negativiteitsbeperkingen op de factoren typisch een betere benadering geeft van een matrix M dan de benadering gevonden met de klassieke niet-negatieve matrix ontbinding. We stellen voor om de niet-negatieve matrixontbinding zonder negativiteitsbeperkingen op de factoren op te lossen met de optimalisatiemethode van Newton.

Vervolgens merken we op dat het vreemd is dat in de literatuur erg veel aandacht besteed wordt aan de niet-negativiteitsbeperkingen (in essentie een ondergrensbeperking) terwijl er geen aandacht wordt besteed aan bovengrenzen.

De niet-negatieve matrixontbinding zonder niet-negativiteitsbeperkingen op de factoren kan makkelijk worden aangepast zodat ze rekening houdt met onder- en bovengrenzen in plaats van enkel met ondergrenzen.

De geïntroduceerde ontbinding wordt toegepast op het comprimeren van een databank van menselijke aangezichten. Het blijkt dat de niet-negatieve matrixontbinding zonder niet-negativiteitsbeperkingen op de factoren betere resultaten geeft dan andere matrixontbindingstechnieken.

Hoofdstuk 3: Verborgen Markov modellen - Lineair stochastische modellen

In dit hoofdstuk introduceren we verborgen Markov modellen en lineair stochastische modellen en beschouwen tevens het equivalentieprobleem voor beide modelklassen.

Zowel verborgen Markov modellen als lineair stochastische modellen hebben geen ingangen die door de gebruiker kunnen worden gecontroleerd. Ze hebben enkel ruisingangen. Om het uitgangsproces elegant te modelleren maken beide modelklassen gebruik van een onderliggend proces, het toestandsproces. Bij verborgen Markov modellen nemen het uitgangs- en toestandsproces waarden aan uit een eindige verzameling, terwijl beide processen voor lineair stochastische modellen waarden aannemen met een continu bereik.

Verborgen Markov modellen

We maken een onderscheid tussen Mealy en Moore verborgen Markov modellen.

Een Mealy verborgen Markov Model is volledig beschreven door de parameters $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ waarbij $\mathbb{X}$ de toestandsverzameling is en $\mathbb{Y}$ de uitgangsverzameling. De vector $\pi(1)$ bevat de kansverdeling voor de toestand op tijdstip 1

$$\pi_i(1) = P(x(1) = i),$$

terwijl Π de kansen bevat om van de ene toestand naar de andere over te gaan en ondertussen een zeker uitgangssymbool te genereren

$$\Pi_{ij}(\mathtt{y}) = P(x(t+1) = j, y(t) = \mathtt{y} | x(t) = i).$$

Bij een Moore verborgen Markov model is de overgang van de toestand op tijdstip t naar de toestand op tijdstip $t+1$ onafhankelijk van het uitgangssymbool op tijdstip t. Een Moore verborgen Markov model wordt beschreven door $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, \beta, \pi(1))$ waarbij $\mathbb{X}$ en $\mathbb{Y}$ de *toestands-* en *uitgangsverzamelingen* zijn. De matrix $\Pi_{\mathbb{X}}$ bevat de kansen om over te gaan van de ene toestand naar de andere

$$(\Pi_{\mathbb{X}})_{ij} = P(x(t+1) = j | x(t) = i),$$

β bevat de kansen op het genereren van een bepaalde symbool in een bepaalde toestand

$$\beta_i(\mathtt{y}) = P(y(t) = \mathtt{y} | x(t) = i),$$

en $\pi(1)$ tenslotte is gedefinieerd op dezelfde manier dan bij Mealy verborgen Markov modellen

$$\pi_i(1) = P(x(1) = i).$$

De stringkansen gegenereerd door een Mealy verborgen Markov model worden gegeven door

$$\mathcal{P}(\mathbf{u}) \quad = \quad \pi(1)\Pi(\mathbf{u})e, \tag{0.3}$$

waarbij $\mathbf{u} = u_1 u_2 \dots u_{|\mathbf{u}|} \in \mathbb{Y}^*$ en waarbij $\Pi(\mathbf{u}) := \Pi(u_1)\Pi(u_2)\dots\Pi(u_{|\mathbf{u}|})$.

In het positieve realisatieprobleem voor verborgen Markov modellen zijn de stringkansen van alle eindige uitgangsstrings gegeven en is het de bedoeling om een verborgen Markov model te vinden dat deze string kansen genereert. Dit probleem is erg moeilijk omwille van de niet-negativiteitsbeperkingen op de systeemmatrices $\pi(1)$ en $\Pi(\mathtt{y}), \mathtt{y} \in \mathbb{Y}$. In dit proefschrift beschouwen we twee afgezwakte versies van het realisatieprobleem: het benaderende positieve realisatieprobleem (Hoofdstuk 5) en het quasi-realisatieprobleem (Hoofdstuk 4). In het quasi-realisatieprobleem voor verborgen Markov modellen wordt dezelfde probleemstelling beschouwd als in het realisatieprobleem, maar zonder negativiteitsbeperkingen op de systeemmatrices. Een model waarbij de systeemmatrices negatieve waarden kunnen aannemen wordt een quasi-HMM genoemd. Een quasi-HMM wordt beschreven door $(\mathbb{Q}, \mathbb{Y}, A, c, b)$. Op het eerste zicht lijkt een quasi-HMM weinig nut te hebben omdat de kansen om over te gaan van de ene toestand naar de andere en om uitgangssymbolen te genereren negatieve waarden kunnen aannemen. We tonen echter aan dat heel wat schattingsproblemen voor verborgen Markov modellen kunnen worden opgelost gebruik makende van quasi-HMMs in plaats van positieve HMMs.

In de thesis beschrijven we een test om na te gaan of een quasi-Mealy verborgen Markov model minimaal is (i.e. of er geen ander quasi-Mealy HMM bestaat dat dezelfde stringkansen genereert, maar dat een kleiner aantal toestanden heeft) en een procedure om een minimaal quasi-Mealy verborgen Markov model te vinden dat equivalent is aan een gegeven niet-minimaal Mealy HMM.

Equivalentie van verborgen Markov modellen

Voor een gegeven verborgen Markov model kan steeds een equivalent model bekomen worden door een permutatie van de toestanden door te voeren. Typisch zijn er echter heel wat meer equivalente modellen mogelijk. We bespreken het equivalentieprobleem voor verborgen Markov modellen en maken daarbij een onderscheid tussen positieve Mealy HMMs, quasi-Mealy HMMs en Moore HMMs.

Voor een minimaal quasi-Mealy HMM $(\mathbb{Q}, \mathbb{Y}, A, c, b)$, kan worden aangetoond dat alle equivalente modellen gegeven worden door $(\mathbb{Q}, \mathbb{Y}, TAT^{-1}, cT^{-1}, Tb)$ waarbij T een reguliere matrix is. Om na te kijken of twee minimale positieve Mealy HMMs equivalent zijn, berekent men eerst voor elk van beide positieve

HMMs een equivalent minimaal quasi-HMM. Indien beide quasi-HMMs nu equivalent zijn, kan worden besloten dat de positieve HMMs equivalent zijn.

We tonen aan dat minimale Moore verborgen Markov modellen die minimaal zijn als een quasi-Mealy HMM enkel triviale equivalenten hebben. Dit wil zeggen dat men enkel een equivalent model kan bekomen door een permutatie van de toestanden door te voeren. Minimale Moore modellen die niet-minimaal zijn als een quasi-Mealy HMM, hebben wel equivalenten die niet bekomen worden door permutatie van de toestanden.

Lineair stochastische modellen

Een Linear Stochastisch Model (LSM) (A, C, P, Q, R, S) wordt gedefinieerd door de volgende differentievergelijkingen

$$\begin{aligned} x(t+1) &= Ax(t) + w(t), \\ y(t) &= Cx(t) + v(t), \end{aligned}$$

waarbij het uitgangsproces y waarden aanneemt in de uitgangsruimte $\mathbb{R}^p$ en het toestandsproces x in de toestandsruimte $\mathbb{R}^n$ waarbij n de orde van het model wordt genoemd. De toevalsvariabelen $w(t)$ en $v(t)$ zijn witte-ruisvariabelen met gemiddelde waarde gelijk aan 0 en met covariantiematrix

$$E(\begin{bmatrix} w(p) \\ v(p) \end{bmatrix} \begin{bmatrix} w(q)^\top & v(q)^\top \end{bmatrix}) = \begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} \delta(p, q).$$

Het toestandsproces wordt stationair verondersteld met als covariantie $P := E(x(t)x(t)^\top)$.

De autocovariantiesequentie $\Lambda(0), \Lambda(1), \Lambda(2), \ldots$ gegenereerd door een lineair stochastisch model (A, C, P, Q, R, S) worden gegeven door

$$\begin{aligned} \Lambda(0) &= CPC^\top + R, \\ \Lambda(t) &= CA^{t-1}G, \end{aligned}$$

waarbij G gedefinieerd is als $G := E(x(t+1)y(t)^\top)$ en berekend als $G = APC^\top + S$. In het realisatieprobleem voor lineair stochastische modellen zijn de autocovarianties gegeven en is het de bedoeling om een lineair stochastisch model te vinden dat deze autocovarianties genereert.

Equivalentie van lineair stochastische modellen

Voor een gegeven lineair stochastisch model (A, C, P, Q, R, S) kan steeds een equivalent model bekomen worden door een basistransformatie in de toestandsruimte door te voeren. Het equivalente model wordt dan gegeven door $(TAT^{-1}, CT^{-1}, TPT^\top, TQT^\top, R, TS)$ waarbij T een reguliere matrix is. Deze transformatie is het analoge van de permutatie van de toestanden die steeds mogelijk is bij verborgen Markov modellen. Net zoals er bij verborgen Markov modellen meer equivalenten mogelijk zijn, zo zijn er ook meer equivalenten mogelijk voor lineair stochastische modellen. Er werd aangetoond dat voor gegeven A, C, G en $\Lambda(0)$, iedere $P = P^\top \succeq 0$ die voldoet aan

$$\left[\begin{array}{c|c} P - APA^\top & G - APC^\top \\ \hline G^\top - CPA^\top & \Lambda(0) - CPC^\top \end{array}\right] \succeq 0,$$

waarbij $X \succeq 0$ betekent dat X niet-negatief definiet is, aanleiding geeft tot een equivalent model $(A, C, P, P - APA^\top, \Lambda(0) - CPC^\top, G - APC^\top)$.

Voor een minimaal Moore lineair stochastisch model (i.e. een LSM waarbij $S = 0$) dat minimaal is als een Mealy LSM, tonen we aan dat de enige mogelijke equivalenten gevormd worden door een basisverandering in de toestandsruimte door te voeren. Deze stelling is volledig analoog aan de stelling voor verborgen Markov modellen. Indien het minimaal Moore LSM niet minimaal is als een Mealy LSM, bestaan er equivalenten die niet bekomen worden door een basisverandering door te voeren in de toestandsruimte.

Hoofdstuk 4: Quasi-realisatie voor verborgen Markov modellen

In het realisatieprobleem voor verborgen Markov modellen zijn de stringkansen van alle eindige uitgangsstrings gegeven en is het de bedoeling om een verborgen Markov model te vinden dat deze string kansen genereert (Vergelijking (0.3)). Dit is een erg moeilijk probleem omwille van het feit dat de systeemmatrices $\pi(1)$ en $\Pi(\mathrm{y}), \mathrm{y} \in \mathbb{Y}$ niet-negatief moeten zijn. In dit hoofdstuk beschouwen we het quasi-realisatieprobleem, identiek hetzelfde probleem maar dan zonder de niet-negativiteitsvereisten op de systeemmatrices.

In het exacte quasi-realisatieprobleem zijn een *oneindig* aantal *exacte* stringkansen gegeven. In het partiële quasi-realisatieprobleem aan de andere kant zijn slechts een eindig aantal exacte stringkansen gegeven en is het de bedoeling een quasi-HMM te vinden dat deze stringkansen genereert. In het benaderende quasi-realisatieprobleem zijn een eindig aantal benaderende stringkansen gegeven en is het de bedoeling een quasi-model te vinden dat deze stringkansen benaderend realiseert. We bespreken nu achtereenvolgens de drie quasi-realisatieproblemen.

Exacte quasi-realisatie

Als eerste stap naar de oplossing van het exacte quasi-realisatieprobleem voor verborgen Markov modellen wordt een dubbel oneindige matrix opgebouwd die de gegeven stringkansen bevat. Deze matrix wordt de *(veralgemeende) Hankelmatrix* genoemd en is gedefinieerd als

$$\mathfrak{H}_{ij} := \mathcal{P}(\mathbf{u}_i \mathbf{v}_j).$$

Voor het geval waar $\mathbb{Y} = \{0, 1\}$, ziet deze matrix eruit als

$$\mathfrak{H} = \left[\begin{array}{c|cc|cccc|c}
1 & \mathcal{P}(0) & \mathcal{P}(1) & \mathcal{P}(00) & \mathcal{P}(01) & \mathcal{P}(10) & \mathcal{P}(11) & \ldots \\
\hline
\mathcal{P}(0) & \mathcal{P}(00) & \mathcal{P}(01) & \mathcal{P}(000) & \mathcal{P}(001) & \mathcal{P}(010) & \mathcal{P}(011) & \ldots \\
\mathcal{P}(1) & \mathcal{P}(10) & \mathcal{P}(11) & \mathcal{P}(100) & \mathcal{P}(101) & \mathcal{P}(110) & \mathcal{P}(111) & \ldots \\
\hline
\mathcal{P}(00) & \mathcal{P}(000) & \mathcal{P}(001) & \mathcal{P}(0000) & \mathcal{P}(0001) & \mathcal{P}(0010) & \mathcal{P}(0011) & \ldots \\
\mathcal{P}(10) & \mathcal{P}(100) & \mathcal{P}(101) & \mathcal{P}(1000) & \mathcal{P}(1001) & \mathcal{P}(1010) & \mathcal{P}(1011) & \ldots \\
\mathcal{P}(01) & \mathcal{P}(010) & \mathcal{P}(011) & \mathcal{P}(0100) & \mathcal{P}(0101) & \mathcal{P}(0110) & \mathcal{P}(0111) & \ldots \\
\mathcal{P}(11) & \mathcal{P}(110) & \mathcal{P}(111) & \mathcal{P}(1100) & \mathcal{P}(1101) & \mathcal{P}(1110) & \mathcal{P}(1111) & \ldots \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}\right].$$

Er kan worden aangetoond dat de stringkansen representeerbaar zijn met een quasi-HMM als en slechts als de rang van de veralgemeende Hankelmatrix eindig is. De rang van de Hankelmatrix is gelijk aan de minimale orde van een quasi-HMM. Aan de hand van een minimale ontbinding van de Hankelmatrix kan nu een algoritme ontworpen worden dat een quasi-realisatie berekent horende bij de gegeven stringkansen.

Partiële quasi-realisatie

In het partiële realisatieprobleem zijn stringkansen voor strings tot lengte t gegeven en is het de bedoeling een quasi-HMM te vinden dat deze stringkansen genereert. Er kan worden aangetoond dat het partiële quasi-realisatieprobleem steeds oplosbaar is. Indien de gegeven stringkansen aan een bepaalde voorwaarde voldoen (de zogenaamde rangconditie), kan een minimale partiële quasi-realisatie bekomen worden door het exacte quasi-realisatie algoritme toe te passen. Onder diezelfde rangconditie is het zo dat een oplossing voor het minimale partiële quasi-realisatieprobleem uniek is tot op een equivalentietransformatie na.

Benaderende quasi-realisatie

In het benaderende quasi-realisatieprobleem zijn benaderende stringkansen gegeven van strings tot lengte t. Indien we de stringkansen exact willen realiseren zal typisch een model van hoge orde nodig zijn. Het is echter beter om een goed lage orde model te maken, eerder dan de stringkansen exact proberen te realiseren.

We stellen een eerste methode voor die een lage-rangbenadering maakt van de blok in de Hankelmatrix die de strings van lengte t bevat. We houden er ook rekening mee dat de benaderende stringkansen consistent en stationair zijn. Op die manier stijgt de rang van de totale Hankelmatrix niet indien we de blok die de strings van lengte t bevat, weer uitbreiden naar een volledige Hankelmatrix. Voor de lage-rangbenadering maken we gebruik van de niet-negatieve matrixontbinding zonder niet-negativiteitsbeperkingen op de factoren (geïntroduceerd in Hoofdstuk 2).

De tweede methode die we voorstellen berekent eerst een quasi-realisatie van volle orde die gebalanceerd is. Doordat de quasi-realisatie gebalanceerd is kan op een eenvoudige manier een quasi-realisatie van gereduceerde orde bekomen worden.

Hoofdstuk 5: Positieve realisatie voor verborgen Markov modellen

Zoals reeds hoger vermeld is het exacte positieve realisatieprobleem voor verborgen Markov modellen een erg moeilijk oplosbaar probleem. In Hoofdstuk 4 beschouwden we een eerste afgezwakte versie van het positieve realisatieprobleem: het quasi-realisatieprobleem. Voor sommige toepassingen is het echter noodzakelijk om toch over een positieve realisatie beschikken. Ook indien men een fysische interpretatie wil geven aan de modelparameters is het noodzakelijk om over een positieve realisatie te beschikken. In dit hoofdstuk wordt daarom het benaderende partiële realisatieprobleem beschouwd.

We beschouwen eerst het speciale geval waar het de bedoeling is om een Moore verborgen Markov model te bekomen voor gegeven stringkansen van strings tot lengte twee. Om dit probleem op te lossen definiëren we de matrix P als volgt

$$P = \begin{bmatrix} \mathcal{P}(\mathrm{y}_1\mathrm{y}_1) & \mathcal{P}(\mathrm{y}_1\mathrm{y}_2) & \dots & \mathcal{P}(\mathrm{y}_1\mathrm{y}_{|\mathbb{Y}|}) \\ \mathcal{P}(\mathrm{y}_2\mathrm{y}_1) & \mathcal{P}(\mathrm{y}_2\mathrm{y}_2) & \dots & \mathcal{P}(\mathrm{y}_2\mathrm{y}_{|\mathbb{Y}|}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{P}(\mathrm{y}_{|\mathbb{Y}|}\mathrm{y}_1) & \mathcal{P}(\mathrm{y}_{|\mathbb{Y}|}\mathrm{y}_2) & \dots & \mathcal{P}(\mathrm{y}_{|\mathbb{Y}|}\mathrm{y}_{|\mathbb{Y}|}) \end{bmatrix}.$$

Indien de stringkansen afkomstig zijn van een Moore verborgen Markov model $(\mathbb{Y}, \mathbb{X}, \Pi_{\mathbb{X}}, \beta, \pi(1))$, dan geldt er dat

$$P = B^\top \operatorname{diag}(\pi(1))\Pi_{\mathbb{X}}B,$$

waarbij $B = \begin{bmatrix} \beta(\mathrm{y}_1) & \beta(\mathrm{y}_2) & \dots & \beta(\mathrm{y}_{|\mathbb{Y}|}) \end{bmatrix}$.

Het Moore realisatieprobleem voor stringkansen van strings tot lengte 2 bestaat er nu in om de gegeven matrix P te benaderen met een product van de vorm $B^\top \operatorname{diag}(\pi(1))\Pi_{\mathbb{X}}B$. Dit probleem kan worden opgelost aan de hand van de gestructureerde niet-negatieve matrixontbinding (geïntroduceerd in Hoofdstuk 2).

Het algemene Moore realisatieprobleem of het algemene Mealy realisatieprobleem kan worden aangepakt door de methode voor het Moore geval met $t = 2$ te veralgemenen. Het is dan ook niet verwonderlijk dat de iteratieve update formules die we afleiden om dit probleem op te lossen een veralgemening zijn van de iteratieve formules die worden gebruikt voor het oplossen van de gestructureerde niet-negatieve matrixontbinding.

Hoofdstuk 6: Identificatie voor verborgen Markov modellen

Het identificatieprobleem bestaat erin een model te maken vanuit gegeven uitgangsmetingen $y_1y_2 \dots y_T$. Voor linear stochastische modellen kunnen de identificatiemethodes in twee groepen worden opgedeeld. Enerzijds zijn er de *predictiefoutmethodes* en anderzijds *deelruimtemethodes.* De eerste groep van methodes is gebaseerd op optimalisatie. De tweede groep van methodes bepaalt, aan de hand van technieken uit de numerieke lineaire algebra, de toestandssequentie rechtstreeks uit de uitgangsdata. Vervolgens worden de systeemmatrices geschat vertrekkende van de uitgangssequentie en de bekomen toestandssequentie.

Identificatie voor verborgen Markov modellen wordt tot op heden opgelost aan de hand van het Baum-Welch algoritme. Deze methode maakt gebruik van optimalisatie en kan beschouwd worden als de tegenhanger van de predictiefoutmethodes. In dit hoofdstuk beschrijven we een methode die analoog is aan de deelruimtemethode voor lineair stochastische modellen. Deze methode schat de toestandssequentie rechtstreeks uit de uitgangsdata en bepaalt vervolgens de systeemmatrices uit de bekomen toestandssequentie en de gegeven uitgangssequentie.

Om de door deelruimte geïspireerde identificatiemethode voor verborgen Markov modellen uit te leggen, is er eerst nood aan de definitie van twee matrices: de toestandsverdelingsmatrix en de volgende-toestandsverdelingsmatrix. De *toestandsverdelingsmatrix* $\tilde{X}_{i_1} \in [0,1]^{(T-i_1)\times|\mathbb{X}|}$ is gedefinieerd als

$$\tilde{X}_{i_1} := \begin{bmatrix} \tilde{x}(i_1+1) \\ \tilde{x}(i_1+2) \\ \vdots \\ \tilde{x}(T) \end{bmatrix},$$

waarbij $\tilde{x}_i(t) := P(x(t) = i|y(t-i_1,...,t-1) = y_{t-i_1}...y_{t-1})$. De *volgende-toestandsverdelingsmatrix* $\tilde{X}^+_{i_1+1} \in [0,1]^{(T-i_1)\times|\mathbb{X}|}$ is gedefinieerd als

$$\tilde{X}^+_{i_1+1} = \begin{bmatrix} \tilde{x}^+(i_1+2) \\ \tilde{x}^+(i_1+3) \\ \vdots \\ \tilde{x}^+(T+1) \end{bmatrix},$$

waarbij $\tilde{x}^+_i(t+1) := P(x(t+1) = i|y(t-i_1,...,t-1) = y_{t-i_1}...y_{t-1})$.

De toestandsverdelingsmatrix en de volgende-toestandsverdelingsmatrix voor een gegeven uitgangssequentie $y_1y_2\dots y_T$ kunnen berekend indien de matrices $\mathfrak{H}_{(i_1+1,i_2+1)}$ en $\mathfrak{H}_{(i_1+2,i_2+1)}$ van het onderliggende HMM en een niet-negatieve ontbinding van deze matrices in $\mathfrak{H}_{(i_1+1,i_2+1)} = VH$ en $\mathfrak{H}_{(i_1+2,i_2+1)} = WH$ gegeven zijn. Vervolgens tonen we aan dat de matrices $\mathfrak{H}_{(i_1+1,i_2+1)}$ en $\mathfrak{H}_{(i_1+2,i_2+1)}$ kunnen geschat worden uit de gegeven uitgangsstring en dat hun niet-negatieve ontbinding kan benaderd worden aan de hand van de niet-negatieve matrixontbinding (Hoofdstuk 2). Resulterend bekomen we een methode om de toestandsverdelingsmatrix en de volgende-toestandsverdelingsmatrix rechtstreeks uit de gegeven uitgangsstring te schatten.

We tonen aan dat de systeemmatrices $\Pi(\mathtt{y}), \mathtt{y} \in \mathbb{Y}$, kunnen berekend worden uit de toestandsverdelingsmatrix en de volgende-toestandsverdelingsmatrix door het oplossen van een kleinste-kwadratenprobleem. De evenwichtstoestandverdeling kan worden berekend als de linkereigenvector bij eigenwaarde 1 van de matrix $\sum_{\mathtt{y}\in\mathbb{Y}} \Pi(\mathtt{y})$.

Hoofdstuk 7: Recursief schatten met quasi-HMMs

Eens een verborgen Markov model of een quasi-HMM bekomen is, kan men het model gebruiken voor allerlei schattingsproblemen. We maken een onderscheid tussen het toestandsschattingprobleem en het uitgangsschattingsprobleem. In het toestandsschattingsprobleem zijn metingen van de uitgang gegeven tot op tijdstip τ en is het de bedoeling om de toestand te schatten op tijdstip t. In het uitgangsschattingsprobleem worden verborgen Markov modellen beschouwd met twee uitgangsprocessen y en z. Metingen van de uitgang y zijn gegeven tot op tijdstip τ en het is de bedoeling om de tweede uitgang te schatten op tijdstip t. Indien $t < \tau$ spreken we van een smoothingprobleem, indien $t = \tau$ van een filteringprobleem en indien $t > \tau$ van een predictieprobleem.

In dit hoofdstuk worden recursieve algoritmes ontwikkeld voor het oplossen van de verschillende toestands- en uitgangsschattingsproblemen. We tonen aan dat de formules voor het oplossen van deze problemen dezelfde blijven indien een quasi-HMM gegeven is in plaats van een positief HMM. Deze observatie heeft belangrijke gevolgen in praktische toepassingen. Indien men een verborgen Markov model wil identificeren met als doel om het te gebruiken voor schattingsproblemen, dan is het niet nodig om een positief verborgen Markov model te bepalen, maar het volstaat om een quasi-HMM te vinden. Dit heeft een aantal voordelen. Vooreerst is het bepalen van een quasi-HMM een makkelijker probleem dan het bepalen van een positief verborgen Markov model. Vervolgens is de orde van een quasi-HMM typisch kleiner dan de orde van een equivalent positief HMM. Dit maakt dat de berekeningen voor het oplossen van schattingsproblemen minder complex.

Een geschakeld HMM bestaat uit twee of meerdere verborgen Markov modellen waartussen op zekere tijdstippen geschakeld wordt. Met behulp van de schattingsmethodes uit dit hoofdstuk kan bepaald worden op welke tijdstippen welk verborgen Markov model actief was. Deze methode kan worden toegepast om motieven te zoeken in DNA sequenties. DNA is een nucleïnezuur dat alle genetische instructies bevat die gebruikt worden voor het ontwikkelen en functioneren van alle gekende levende organismen. DNA vormt een dubbele helix van complementaire nucleotidesequenties. De nucleotidesequenties bestaan uit een opeenvolging van 4 nucleotiden: adenine (A), cytosine (C), guanine (G) en thymine (T). Bepaalde delen van het DNA reguleren de vorming van bepaalde proteïnes. En stap in het proces van DNA naar proteïne is de binding van een zekere transcriptiefactor met het DNA. Er werd aangetoond dat er een zekere complementariteit moet bestaan tussen de transcriptiefactor en een deel van het DNA voordat de binding kan plaatsvinden. Een model voor een deel van het DNA waar een binding met een transcriptiefactor kan plaatsvinden wordt een motief genoemd. Indien we nu een verborgen Markov model van een motief hebben en een verborgen Markov model van de achtergrond dan kunnen de schattingsmethodes van dit hoofdstuk gebruikt worden voor het zoeken naar motieven in DNA-sequenties.

Hoofdstuk 8: Besluit

In dit proefschrift worden verborgen Markov modellen en lineair stochastische modellen bestudeerd. Hoewel beide modelklassen erg gelijkaardig zijn, zijn er voor verborgen Markov modellen nog een heel aantal theoretische problemen onopgelost terwijl de overeenkomstige problemen voor lineair stochastische modellen grotendeels opgelost zijn. De eerste grote doelstelling van dit doctoraat is bijdragen leveren aan de open theoretische problemen over verborgen Markov modellen. Bij het oplossen van deze problemen kan inspiratie worden gezocht in de oplossing van de overeenkomstige problemen voor lineair stochastische problemen. Bij het oplossen van problemen over lineair stochastische modellen wordt gebruik gemaakt van de singuliere-waardenontbinding. Het blijkt dat voor het oplossen van de overeenkomstige problemen over verborgen Markov modellen nood is aan de niet-negatieve matrixontbinding en varianten op deze ontbinding. Het ontwikkelen van varianten op de niet-negatieve matrixontbinding vormt de tweede grote doelstelling van dit proefschrift.

In dit proefschrift wordt het gestructureerde niet-negatieve matrixfactorisatieprobleem geïntroduceerd en worden iteratieve formules voorgesteld om dit probleem op te lossen. Ook wordt de niet-negatieve matrixontbinding zonder niet-negativiteitsbeperkingen op de factoren voorgesteld en ook hier wordt een algoritme gegeven om dit probleem op te lossen. Beide methodes werden ontwikkeld om problemen over verborgen Markov modellen op te lossen, maar hebben toepassingen op zichzelf, los van het onderzoek naar verborgen Markov modellen.

Een belangrijk probleem aangaande verborgen Markov modellen is het realisatieprobleem: gegeven stringkansen, vind een bijhorend verborgen Markov model. Dit probleem is moeilijk oplosbaar omwille van niet-negativiteitsvereisten op de systeemmatrices van een verborgen Markov model. In dit proefschrift worden twee afgezwakte versies van het realisatieprobleem besproken. De eerste afgezwakte versie is het quasi-realisatieprobleem waar de niet-negativiteitsbeperkingen op de systeemmatrices worden weggelaten. Het tweede afgezwakte probleem is het benaderende positieve realisatieprobleem waar het niet de bedoeling is om een HMM te bekomen dat de stringkansen exact realiseert, maar waar het voldoende is dat het HMM de stringkansen benaderend realiseert.

In het identificatieprobleem voor verborgen Markov modellen is een uitgangsstring gegeven en is het de bedoeling om een model van die string te bepalen. Voor dit probleem is het Baum-Welch algoritme beschikbaar, een identificatiemethode gebaseerd op optimalisatie. In dit proefschrift wordt een methode voorgesteld die geïnspireerd is op deelruimte-identificatie voor lineair stochastische modellen. In deze methode wordt de toestandssequentie rechtstreeks uit de data geschat en vervolgens worden de systeemmatrices bepaald uit de toestands- en uitgangssequenties.

Eens een verborgen Markov model geïdentificeerd is, kan het gebruikt worden voor het oplossen van schattingsproblemen. In toestandsschattingsproblemen is de uitgang gegeven tot op een zeker tijdstip en is het de bedoeling de toestand op een ander tijdstip te schatten. In uitgangsschattingsproblemen beschouwt men verborgen Markov modellen met twee uitgangsprocessen. Men veronderstelt dat de eerste uitgang gegeven is tot op een zeker tijdstip en het is de bedoeling de tweede uitgang te schatten op een ander tijdstip. In dit proefschrift werd aangetoond dat het voor uitgangsschattingsproblemen volstaat om een quasi-HMM te hebben in plaats van een positief verborgen Markov model. Deze observatie heeft een aantal voordelen: ten eerste kan een quasi-HMM makkelijker bekomen worden uit data en ten tweede heeft een quasi-HMM typisch een lagere orde dan een positief verborgen Markov model zodat de schattingsberekeningen minder complex worden.

Chapter 1

Introduction

1.1 Motivation and objectives

A system is a physical, economical, biological, industrial, technical, ... phenomenon that interacts with its environment. The behavior of systems is usually analyzed on the basis of a mathematical model. A mathematical model describes the relation between certain variables of the system as a function of time. Typically, the variables are divided into inputs and outputs. Some of the inputs can be controlled by the user, others not. The outputs are a consequence of the inputs and can not be controlled directly. Figure 1.1 schematically shows a mathematical model with control inputs u, disturbance inputs v, and outputs y. Models are highly useful in situations where experimenting with the real system is too expensive, too dangerous, or technically impossible.

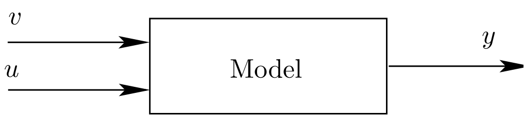


Figure 1.1: *A mathematical model with input u, disturbance input v and output y. The user can control u, but not v.*

The first main objective of this thesis is the detailed investigation of one specific class of mathematical models: discrete time *hidden Markov models*. Although hidden Markov models have been used in many applications (speech processing [60, 86], computational biology, such as identifying the genes of an organism from its DNA [31,68,90] and classifying proteins into a small number of families [67]), many theoretical questions concerning the models are open until now. Hidden Markov models are closely related to discrete time *linear stochastic models* for which the theoretical research has attained a certain level of maturity.

When solving theoretical problems concerning hidden Markov problem, we can find inspiration in the solution of the corresponding problem for linear stochastic systems.

The solution to most of the theoretical problems for linear stochastic models, in one way or another, makes use of the singular value decomposition, an important matrix factorization technique from linear algebra. To solve the corresponding questions for hidden Markov models, it turns out that we need another matrix factorization, the nonnegative matrix factorization. Also modified versions of this matrix factorization technique will be needed. Deriving modifications to the nonnegative matrix factorization forms the second important objective of this thesis. We try to keep the modifications to the matrix factorizations as general as possible such that they can be used on their own, apart from the hidden Markov research.

We now describe the research objectives concerning hidden Markov models (Section 1.1.1) and concerning matrix factorizations (Section 1.1.2) into more detail.

1.1.1 Hidden Markov models - linear stochastic models

The first main objective of this thesis is to solve some open theoretical questions concerning hidden Markov models. Hints for the solution can be found in the solution of the corresponding problem for linear stochastic models. In Section 1.1.1.1, we introduce hidden Markov models and linear stochastic models. Subsequently, in Section 1.1.1.2, Section 1.1.1.3 and Section 1.1.1.4, we describe the different theoretical questions that will be considered.

1.1.1.1 Hidden Markov models - linear stochastic models

In this section we introduce hidden Markov models and linear stochastic models and describe some theoretical questions for both model classes. These theoretical questions are further explained in the next sections.

In this thesis we consider hidden Markov models and linear stochastic models that do not have inputs that can be controlled by the user. To model the output process effectively, both models make use of an internal process called the *state process*. The state process may or may not have a clear physical meaning, but is of conceptual relevance. The state and output processes of a hidden Markov model take values in a finite set while the state and output process of linear stochastic models take values with a continuous range of values.

A (positive) Hidden Markov Model[1] (HMM) $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ with state set $\mathbb{X}$ of cardinality $|\mathbb{X}|$ and output set $\mathbb{Y}$ of cardinality $|\mathbb{Y}|$ is completely described by the distribution of the initial state $\pi(1)$, and $\Pi_{ij}(\mathtt{y})$, the probability of going from state i to state j and producing output symbol $\mathtt{y}$ from the output set

[1]When refering to a hidden Markov model, the word "positive" may be added to make a distinction with a "quasi" hidden Markov model (defined further). If "positive" or "quasi" is omitted, it should be clear from the context, which of both models is mentioned.

$\mathbb{Y}$. The cardinality of the state set $|\mathbb{X}|$ is called the *order* of the model. More formally, $\pi(1)$ is a vector in $\mathbb{R}_+^{1\times|\mathbb{X}|}$ defined by

$$\pi_i(1) = P(x(1) = i),$$

and Π is a mapping from the output space $\mathbb{Y}$ to matrices in $\mathbb{R}_+^{|\mathbb{X}|\times|\mathbb{X}|}$ defined as

$$\Pi_{ij}(\mathtt{y}) = P(x(t+1) = j, y(t) = \mathtt{y}|x(t) = i).$$

A graphical representation of a hidden Markov model with three states is given in Figure 1.2.

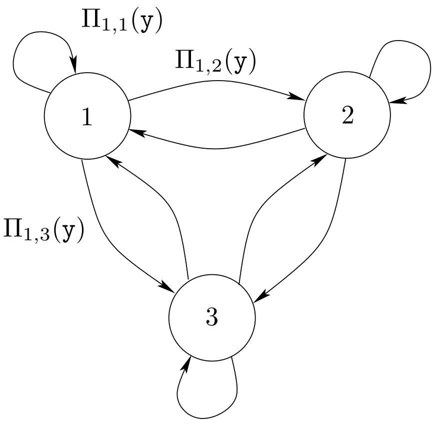


Figure 1.2: *Graphical representation of a hidden Markov model with state set* $\{1, 2, 3\}$*. The model starts in one of the three states according to the distribution* $\pi(1)$*, with* $\pi_i(1) = P(x(1) = i)$*. At every time instant, the model switches between the states and produces an output symbol from the output set* $\mathbb{Y}$ *according to* $\Pi_{ij}(\mathtt{y}) = P(x(t+1) = j, y(t) = \mathtt{y}|x(t) = i)$*.*

A Linear Stochastic Model (LSM) (A, C, Q, R, S) is defined by the following set of difference equations

$$\begin{aligned} x(t+1) &= Ax(t) + w(t), \\ y(t) &= Cx(t) + v(t), \end{aligned} \tag{1.1}$$

where y is the output process taking values in the output space $\mathbb{R}^p$ and x is the state process taking values in the state space $\mathbb{R}^n$ where n is called the *order* of the model. The random variables $w(t)$ and $v(t)$ are zero mean, white vector variables with covariance matrix

$$E(\begin{bmatrix} w(p) \\ v(p) \end{bmatrix} \begin{bmatrix} w(q)^\top & v(q)^\top \end{bmatrix}) = \begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} \delta(p, q),$$

where

$$\begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} \succeq 0,$$

and $\delta(p,q)$ is the Kronecker delta. A graphical representation of the linear stochastic model is given in Figure 1.3.

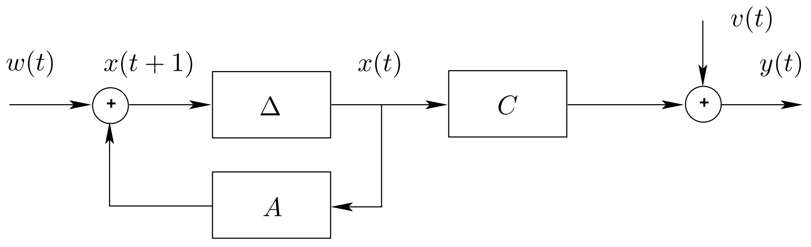


Figure 1.3: *Graphical representation of a linear stochastic model. The vector signal $y(t)$ represents the output, $v(t)$ and $w(t)$ are unknown disturbances. The symbol Δ represents a delay. Note the inherent feedback via the matrix A which represents the dynamics.*

For both hidden Markov models as well as linear stochastic models, one has the theoretical questions below.

- Given a model, derive conditions under which a second model is equivalent (i.e. has the same external behavior) to the given model. In addition, describe the complete set of all equivalent models.
- Given an output sequence of the model, derive the underlying system order and find the model parameters. This problem is called the *identification problem.*
- Given an external description of the model (in terms of an autocovariance sequence, string probabilities (see further)), find an internal description (with states). This problem is called the *realization problem*[2].
- Given the output sequence up to a certain time instant, predict the state and/or output at the next time instants. Problems of this kind are called *estimation problems.*

All questions above have been considered and "solved"[3] for linear stochastic models. However for hidden Markov models, many questions remain unsolved

[2]In this thesis we consider the "weak" realization problem. The weak realization problem aims at modeling the statistics of a process (string probabilities, autocovariances). In the remainder of the thesis, the word "weak" is omitted when referring to the weak realization problem.

[3]With "solved" we mean that there exist algorithms to solve "standard" versions of the above problems. However, there exist special cases where the problems are unsolved and where further research is needed.

until now. This fact is amazing because of the very close relation between hidden Markov models and linear stochastic models and because of the fact that hidden Markov models have been used in many engineering applications. The first important objective of this thesis is to contribute to the above questions for hidden Markov models, starting from the knowledge of the corresponding solution for linear stochastic models. In Section 1.1.1.2, Section 1.1.1.3 and Section 1.1.1.4, we describe this objective into more detail.

1.1.1.2 Realization of hidden Markov models

The *exact realization problem* for linear stochastic models consists of finding a state space model (Equation (1.1)) corresponding to a given autocovariance sequence of the output process. This question consists of three different parts. The first part is the *realizability problem*: under which conditions is an autocovariance sequence representable by a finite-dimensional linear stochastic model. The second part is the *realization problem* itself: given a realizable autocovariance sequence, find a corresponding minimal state space model. The last question is the *equivalence problem*: given a realizable autocovariance sequence, derive *all* corresponding (minimal) state space models. These questions have been considered in [3, 46, 51].

The exact realization problem for hidden Markov models consists of finding a hidden Markov model (i.e. the order $|\mathbb{X}|$ and system matrices $\pi(1)$ and Π) corresponding to given string probabilities of all finite length output strings. Again, the problem can be split up into three parts: the realizability problem, the realization problem itself and the equivalence problem.

In Table 1.1, we schematically show the three steps of the realization problem (both for linear stochastic models as well as for hidden Markov models).

The realization problem is nice from theoretical point of view. However, it supposes an *infinite* amount of *exact* autocovariances/string probabilities to be given. In practice however, only a finite amount of exact autocovariances/string probabilities or a finite amount of approximate autocovariances/string probabilities are given. The partial realization problem finds a model corresponding to a finite number of external parameters and the approximate partial realization problem to a finite number of approximate external parameters. In Table 1.2, the difference between the exact, partial and approximate realization problem is presented.

We now discuss the realization problem for hidden Markov models into more detail, highlighting the open topics and indicating on which topics we will work in this thesis. We explain the link with the realization problem for linear stochastic models. In Table 1.3, we summarize the analogies between the realization problem for linear stochastic models and for hidden Markov models.

The data for the exact linear stochastic realization problem are exact autocovariances. An exact autocovariance sequence is positive real [78]. The data for the exact hidden Markov realization problem are exact string probabilities. Exact string probabilities are positive and in addition fullfill some consistency properties.

Table 1.1: *The realization problem (both for linear stochastic models as well as for hidden Markov models) consists of three steps: the realizability problem, the realization problem itself and the equivalence problem.*

EXACT REALIZATION PROBLEM
Realizability problem *Given:* Autocovariances/string probabilities *Find:* Conditions for realizability by a LSM/HMM
Realization problem *Given:* Realizable autocovariances/string probabilities *Find:* LSM/HMM realizing the autocovariances/string probabilities
Equivalence problem *Given:* Realizable autocovariances/string probabilities *Find:* *All* LSM/HMM realizing the autocovariances/string probabilities

Table 1.2: *Depending on whether a finite or infinite amount of exact or approximate autocovariances/string probabilities are given, a distinction is made between the exact realization problem, the partial realization problem and the approximate realization problem.*

EXACT, PARTIAL AND APPROXIMATE REALIZATION PROBLEMS
Exact realization problem *Given:* *Infinite* amount of *exact* autocovariances/string probabilities
Partial realization problem *Given:* *Finite* amount of *exact* autocovariances/string probabilities
Approximate realization problem *Given:* *Finite* amount of *approximate* autocovariances/string probabilities

An important contraint in the linear stochastic realization problem is that the covariance matrices Q and R of a model need to be positive definite. Only linear stochastic models (A, C, Q, R, S) with Q and R positive definite have physical relevance. For hidden Markov models on the other hand, the system matices $\pi(1)$ and Π need to be elementwise nonnegative. In the realization problem for hidden Markov models only solutions $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ with $\pi(1)$ and

Π elementwise nonnegative are allowed. It will turn out that these constraints are very hard to work with in practice. A relaxed version of the problem, the *quasi realization problem* is defined in the same way as the realization problem but without the nonnegativity constraints on the matrices $\pi(1)$ and Π. A solution to the quasi realization problem is called a *quasi hidden Markov model.* We show that in many practical applications it suffices to have a quasi realization instead of a positive realization (see Section 1.1.1.4).

Concerning the realizability problem, it is shown that an autocovariance sequence is realizable by a linear stochastic model *if and only if* a certain doubly infinite matrix containing the autocovariances, the *Hankel matrix of autocovariances*, has finite rank [3,51]. In that case it is possible to find a linear stochastic model with Q and R positive definite. For hidden Markov models, one also defines a doubly infinite matrix containing the string probabilities, the *Hankel matrix of string probabilities.* It can be shown that the string probabilities are representable by a hidden Markov model *only if* that matrix has finite rank [20, 24, 52]. The rank condition is only a necessary condition for the existence of a realization [39, 50]. Up to now it is an important open problem to derive necessary and sufficient conditions for string probabilities to be representable by a hidden Markov model. For quasi hidden Markov models, it can be shown that the rank condition is a necessary and sufficient condition for string probabilities to be realizable [52, 84].

The solution to the linear stochastic realization problem lies in the factorization of the Hankel matrix containing the autocovariances [3, 51]. For hidden Markov models it is an important open problem to find a hidden Markov model corresponding to given string probabilities [4]. The quasi realization problem for hidden Markov models on the other hand can be solved by decomposing the Hankel matrix of the string probabilities [52, 84].

The equivalence problem for linear stochastic models has been considered in [46]. First of all, an equivalence transformation applied on a given model gives rise to an equivalent model. On the other hand, for a given model with a certain state covariance there exists a convex set of state covariances that give rise to equivalent models. For hidden Markov models, a permutation of the states is the analogue of the equivalence transformation for linear stochastic models. It is a research topic of this thesis to check whether there exist more equivalents then only the ones obtained by permuting the states. For a given quasi hidden Markov model, a permutation of the states gives rise to an equivalent model. Moreover, it is shown in [113], that all equivalent models are linked by a similarity transformation.

The partial realization problem for linear stochastic models has been considered in [54, 63, 98]. The partial quasi realization problem for hidden Markov models is a research topic of this thesis. The partial realization problem for hidden Markov models has been considered in [114]. The approximate partial realization problem for linear stochastic systems has been considered in [71]. The approximate partial quasi realization problem for hidden Markov models is a research topic of this thesis. The approximate partial realization problem for hidden Markov models has been investigated in [47] and is further investigated in this thesis.

Table 1.3: *Analogies between the realization problem for linear stochastic models and the realization problem for hidden Markov models.*

LINEAR STOCHASTIC MODELS	HIDDEN MARKOV MODELS
Autocovariance sequence positive real	**String probabilities** nonnegative/consistent
Linear stochastic model Q, R, S positive definite	**Hidden Markov model** $\pi(1)$ and Π elementwise nonnegative **Quasi hidden Markov model** $\pi(1)$ and Π can be negative
Realizability [3,51] rank Hankel matrix $< \infty$	**Quasi realizability** [52,84] rank Hankel matrix $< \infty$ **Realizability** [20,24,39,50,52] open problem
Realization [3,51] factorize Hankel matrix	**Quasi realization** [52,84] factorize Hankel matrix **Realization** [4] open problem
Equivalence problem [46] -equivalence transform -state covariance in convex set	**Quasi equivalence problem** [113] -permutation of the states -equivalence transform **Equivalence problem** -permutation of states -*this thesis*
Partial realization [54,63,98]	**Partial quasi realization** *this thesis* **Partial realization** [114]
Approximate realization [71]	**Approximate quasi realization** *this thesis* **Approximate realization** [47] *this thesis*

We now summarize the engineering approach that we followed to tackle the positive hidden Markov realization problem. The exact positive realization problem for hidden Markov models is hard to solve. Therefore, we consider two relaxations of the exact positive realization problem (see Figure 1.4). The first is the quasi realization problem where the nonnegativity constraint on the system matrices is omitted. The second is the approximate positive realization problem.

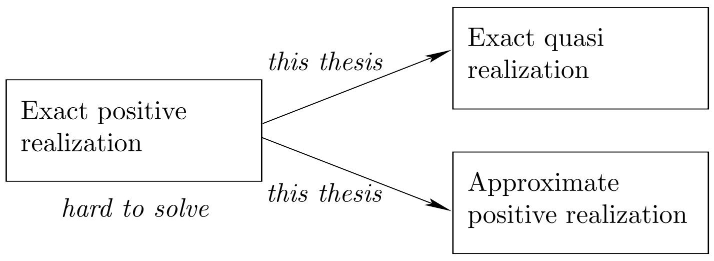
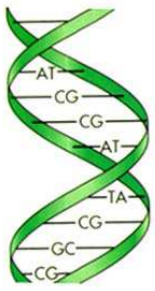


Figure 1.4: *We show the engineering approach that was followed to tackle the positive realization problem. The exact positive relization problem is hard to solve. In this thesis, we consider two relaxations of the exact positive realization problem. The first is the quasi realization problem where the nonnegativity constraint on the system matrices is omitted. The second is the approximate positive realization approach.*

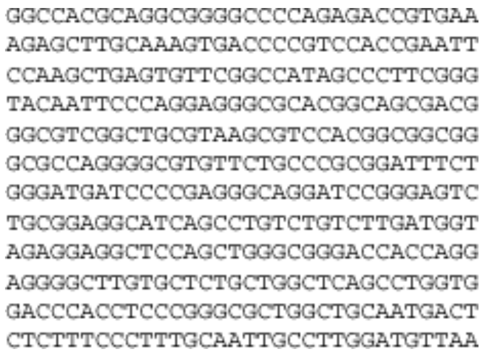
The quasi realization procedure and the approximate positive realization procedure are applied to the modeling of DNA sequences. Desoxyribonucleic acid (DNA) is a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms. DNA forms a double helix of two anti-parallel chains with complementary nucleotide sequences. In Figure 4.3(a), the double DNA helix is schematically shown. The building blocks of the nucleotide sequences are four nucleotides: adenine (A), cytosine (C), guanine (G) and thymine (T). The human genome consists of approximately 3 billion nucleotide pairs. In Figure 4.3(b), an example of a part of a DNA sequence is shown.

1.1.1.3 Identification of hidden Markov models

The identification problem consists of making a model starting from input and output measurements. For linear stochastic models identification methods can be subdivided into two classes. The first class are the *prediction error methods* [75], optimization based methods that minimize the prediction error, the difference between the observed output and the output predicted by the identified model. The second class of methods are the *subspace based methods* [78]. These methods first derive the state sequence directly from output data. In a next step, the system matrices are estimated from the state and output sequence by solving a least squares problem. Subspace based methods make use of numerically stable operations from linear algebra as the singular value



(a)



```
GGCCACGCAGGCGGGCCCCAGAGACCGTGAA
AGAGCTTGCAAAGTGACCCCGTCCACCGAATT
CCAAGCTGAGTGTTCGGCCATAGCCCTTCGGG
TACAATTCCCAGGAGGGCGCACGGCAGCGACG
GGCGTCGGCTGCGTAAGCGTCCACGGCGGCGG
GCGCCAGGGGCGTGTTCTGCCCGCGGATTTCT
GGGATGATCCCCGAGGGCAGGATCCGGGAGTC
TGCGGAGGCATCAGCCTGTCTGTCTTGATGGT
AGAGGAGGCTCCAGCTGGGCGGGACCACCAGG
AGGGGCTTGTGCTCTGCTGGCTCAGCCTGGTG
GACCCACCTCCCGGGCGCTGGCTGCAATGACT
CTCTTTCCCTTTGCAATTGCCTTGGATGTTAA
```

(b)

Figure 1.5: *Desoxyribonucleic acid (DNA) is a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms. DNA forms a double helix of two anti-parallel chains with complementary nucleotide sequences (Subfigure (a)) . The human genome consists of approximately 3 billion nucleotide pairs. In Subfigure (b), an example of a DNA sequence of length 384 is shown.*

decomposition, oblique projections, least squares,... Subspace based methods have the advantage over prediction error methods that they are numerically stable and can be used for identification of MIMO (multiple input, multiple output) models in an elegant way.

Identification for hidden Markov models is usually performed using the Baum-Welch algorithm, an optimization based approach based on maximum likelihood. In [10] results on consistency and asymptotic normality of the maximum likelihood estimator are given, and the conditions for consistency are weakened in [83]. Consistency and asymptotic normality properties of the maximum likelihood estimator are further investigated in [18, 19]. The Baum-Welch approach is based on optimization and we consider it therefore as the analogue of the prediction error methods for linear stochastic systems.

In this thesis, we develop an identification method for hidden Markov models inspired by the ideas of subspace identification. More precisely, the method first estimates the state sequence from the given output sequence and subsequently computes the system matrices from the given output sequence and the obtained state sequence. In the identification method for hidden Markov models the nonnegative matrix factorization plays the role that the SVD plays in subspace identification. In Figure 1.6, we show the relation between the introduced identification method and the existing methods for HMMs and LSMs. In Figure 1.7, we schematically show the difference between both identification methods for hidden Markov models.

We use the subspace inspired identification method to model the changes observed in the DNA of a highly mutating virus: the Human Immunodeficiency

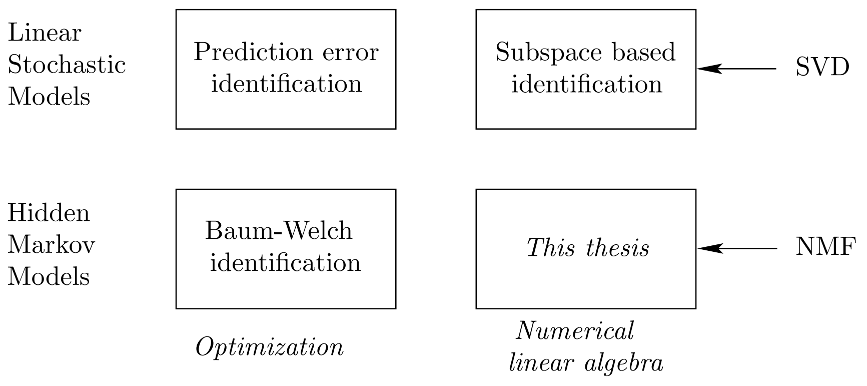


Figure 1.6: *For linear stochastic models the system identification methods can be subdivided into two classes. The first class contains the* prediction error methods*. The second class of methods are formed by the* subspace based methods*. Subspace methods have the advantage that they consist of numerically stable operations (SVD, ...) and that they can be easily used for MIMO identification. For hidden Markov models, up to this moment, identification is performed using the Baum-Welch algorithm. This approach can be considered as the analogue of the prediction error methods for linear stochastic systems. In this thesis we propose an identification method for hidden Markov models inspired by subspace identification of linear stochastic models. In this new identification method, the nonnegative matrix factorization plays the role that the SVD plays in subspace identification.*

Virus (HIV). Mutations are changes to the nucleotide sequence of the genetic material of an organism. Gene mutations take many forms and can result in the loss of complete sections of genes, their duplications or inversions. New segments of DNA can be integrated, or genes can be broken into parts and separated. However, by far the most common type of mutations, called *point mutations*, result from the replacement of a single nucleotides within a gene. In Figure 1.8(a) we illustrate the principle of a point mutation. Radiation, exposure to certain chemicals, and some biological processes can induce mutations in genes. But even in the absence of these influences the genes of all living organisms are subject to mutations. This background mutation rate for cellular organisms is low and consequently can only be observed in organisms that reproduce in enormous numbers, like the HIV virus. In Figure 1.8(b) we schematically show the HIV virus. It has been estimated that every possible single point mutation in the HIV may occur more than 10,000 times a day in an affected person [27]. All these factors contribute to a larger number of *random* changes introduced into the viral particles that eventually lead to structural modifications. It is thus reasonable to assume that the HIV mutational processes are such that

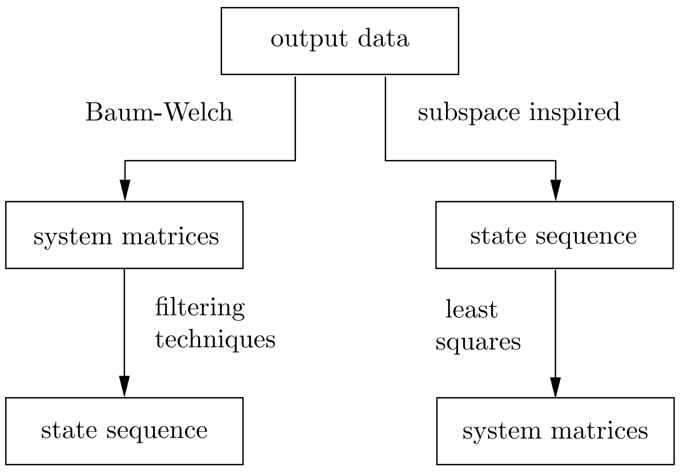


Figure 1.7: *Identification methods aim at constructing models from output data. The left hand side shows the classical approach: first obtain the system matrices, then estimate the states if needed. The right hand side shows the approach inspired by subspace identification for linear stochastic models: first the states are estimated directly from data, then the system matrices are computed.*

the sequences produced randomly traverse through the space of all possible sequences. We use the subspace inspired identification method to obtain a hidden Markov model of the HIV sequences. The quality of the model is assessed on a test dataset. Finally, we explain how the model can be used to predict new viral sequences.

1.1.1.4 Estimation techniques for hidden Markov models

Once a linear stochastic model is obtained, either by realization or by identification, the model can be used for *estimation.* The most important estimation problem is the one where an output sequence up to a certain time instant t is given and the problem is to estimate the state and/or output at time instant $t + 1$. However, many different types of estimation problems can be defined. For linear stochastic systems, estimation problems are solved using Kalman filtering techniques (for a review see [6]).

For hidden Markov models, some estimation techniques for positive hidden Markov models have already been proposed in the literature. In this thesis, it is shown that certain estimation problems can be solved using quasi hidden Markov models instead of positive hidden Markov models. This observation is nice for two reasons. First of all, quasi hidden Markov models are more easy

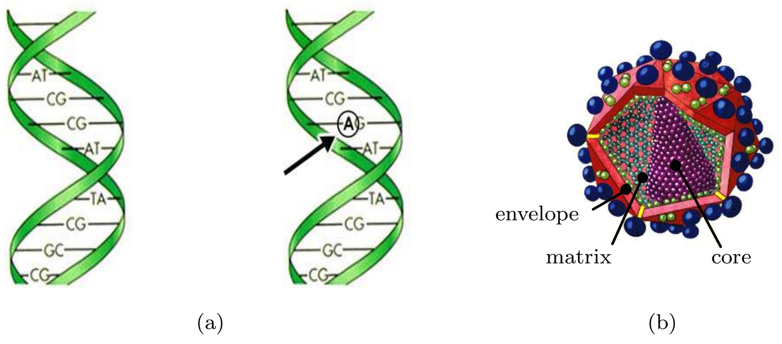


Figure 1.8: *Mutations are changes to the nucleotide sequence of the genetic material of an organism. In Subfigure (a) we illustrate the principle of a point mutation, where a single nucleotides within a gene is replaced (here the C-nucleotide is replaced by the A-nucleotide). The background mutation rate for cellular organisms is low and consequently can only be observed in organisms that reproduce in enormous numbers, like the HIV virus. In Subfigure (b) we schematically show the HIV virus. The core of the HIV virus is surrounded by a matrix composed of a viral protein. This is, in turn, surrounded by the viral envelope.*

to obtain than positive hidden Markov models. In addition, the order of a quasi hidden Markov model is typically smaller than the order of an equivalent positive hidden Markov model.

The proposed estimation techniques for hidden Markov models are used to detect "important parts" in DNA sequences. Certain parts of the DNA (the genes) regulate the formation of certain proteines. One step in the process from DNA to proteine is the binding of a certain transcription factor with the DNA. It has been shown [13], that there must exist a certain complementarity between the transcription factor and a part of the DNA for a binding to take place. A model for a part of the DNA where possibly a binding with a transcription factor can take place is called a *motif*. In this thesis we use the estimation methods for hidden Markov models to detect the positions of motifs in DNA sequences.

1.1.2 Matrix factorizations

The solutions to most of the problems (realization, identification, estimation) concerning linear stochastic models make use of the singular value decomposition. To solve the corresponding problems for hidden Markov models, there is need for another matrix factorization technique, the nonnegative matrix factorization, as well as certain modifications to this decomposition. As explained before, the derivation of new nonnegative matrix factorization techniques forms a second objective of this thesis. As a nice side effect, the matrix factorization algorithms developed in this thesis are useful on their own, apart from the research on hidden Markov models. We introduce the need for two modifications to the nonnegative matrix factorization in Section 1.1.2.1 and Section 1.1.2.2. We first recall the (approximate) nonnegative matrix factorization problem, introduced by Lee and Sueng [72]:

Problem 1.1. *Given $M \in \mathbb{R}_+^{m_1 \times m_2}$ and given $a \in \mathbb{N}$, minimize $D_{KL}(M||VH)$ with respect to V (of size $m_1 \times a$) and H (of size $a \times m_2$), subject to the constraints $V \geq 0$, $H \geq 0$.*

$D_{KL}(A||B)$ is the Kullback-Leibler divergence between two nonnegative matrices and is defined as

$$D_{KL}(A||B) = \sum_{ij}(A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij}).$$

Lee and Sueng propose iterative update formulas to solve Problem 1.1 and prove interesting properties of the formulas.

1.1.2.1 Structured nonnegative matrix factorization with application to object clustering

The (approximate) structured nonnegative matrix factorization problem can be stated as

Problem 1.2. *Given $P \in \mathbb{R}_+^{p \times p}$ and given $a \in \mathbb{N}$, minimize $D_{KL}(P||VAV^\top)$ with respect to V (of size $p \times a$) and A (of size $a \times a$), subject to the constraints $V \geq 0$, $A \geq 0$.*

In this thesis iterative update formulas to solve this factorization problem will be derived.

The need for the structured nonnegative matrix factorization comes from the theoretical research concerning hidden Markov models. However, it turns out that the structured nonnegative matrix factorization has applications that have nothing to do with hidden Markov models. One of these applications is the problem of clustering data points based on their distance matrix. We start with an intuitive motivation to the clustering problem.

In Figure 1.9, we show three types of iris flowers: the setosa type (Figure 1.9(a)), the versicolor type (Figure 1.9(b)) and the virginica type (Figure 1.9(c)). The distinction between the different types can be made based on the size of the sepals and the petals of the flowers (see Figure 1.9(d) for the definition of sepals an petals). Now, suppose a set of flowers is given, as well as measurements of the length and width of their sepals and petals (this data set is available in the SOM-toolbox for Matlab as the file `iris.mat`). The objective is to divide the set into three subsets corresponding to setosa flowers, versicolor flowers and virginica flowers. The problem described here is a typical *clustering problem.* In a clustering problem, one has a number of p *objects* (flowers) of which n *features* (sepal length, sepal width, petal length and petal width) are given and

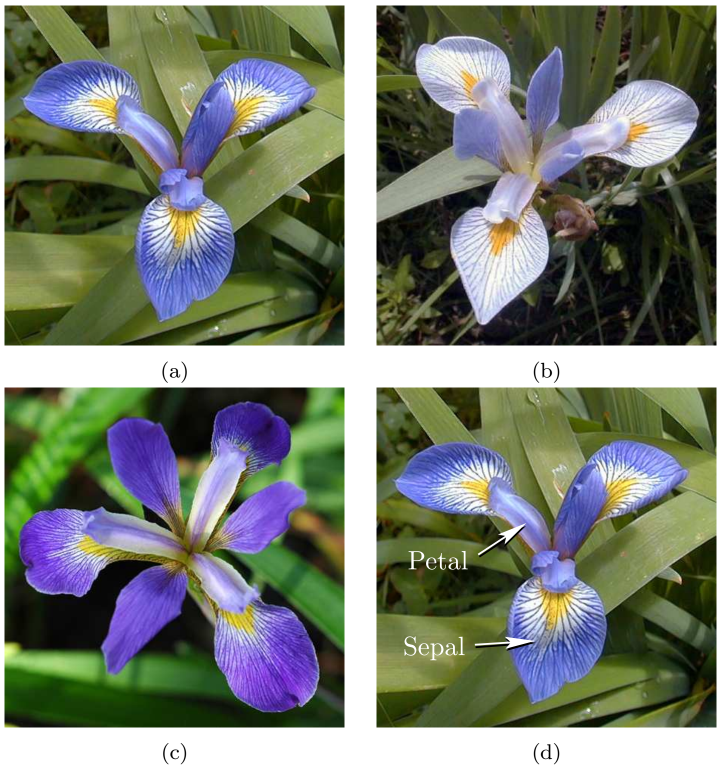


Figure 1.9: *We show three types of iris flowers: the setosa type (Subfigure (a)), the versicolor type (Subfigure (b)) and the virginica type (Subfigure (c)). The distinction between the different types can be made by based on the size of the sepals and the petals of the flowers (Subfigure (d)). Suppose a set of flowers is given, as well as measurements of the length and width of their sepals and petals. The objective is to divide the set into three subsets corresponding to setosa flowers, versicolor flowers and virginica flowers. The problem described here is a typical* clustering problem.

the objective is to find *clusters* of objects that have features values that are *close* to each other.

Typically, the p objects with n features are represented as p points in $\mathbb{R}^n$. Now given a certain distance measure on $\mathbb{R}^n$, the distance matrix P between the p points can be calculated $P_{kl} = D(y_k, y_l)$. Note that P is symmetric and that the diagonal elements of P are equal to 0. In this thesis, it will be shown that the clustering problem of p points with distance matrix P into a clusters, is equivalent to the following matrix factorization problem

$$\begin{array}{ll} \text{minimize} & C(P, VAV^\top) \\ \text{subject to} & V \in \{0,1\}^{p\times a} \\ & Ve = e, \end{array}$$

where $C(X,Y)$ is a distance measure between the matrices X and Y. This problem is hard to solve. However, the problem is close to the structured nonnegative matrix factorization problem considered in this thesis. We show that the structured nonnegative matrix factorization can be used for clustering data points based on their distance matrix. In Figure 1.10, we show the results of our clustering algorithm. Our algorithm finds three clusters of flowers of which the features are "close" to each other. Points that belong to cluster 1 are plotted with o, point belonging to cluster 2 are plotted with $*$, and points belonging to cluster 3 are plotted with $+$. It turns out that cluster 1 corresponds to versicolor flowers, cluster 2 to virginica flowers and cluster 3 to setosa flowers.

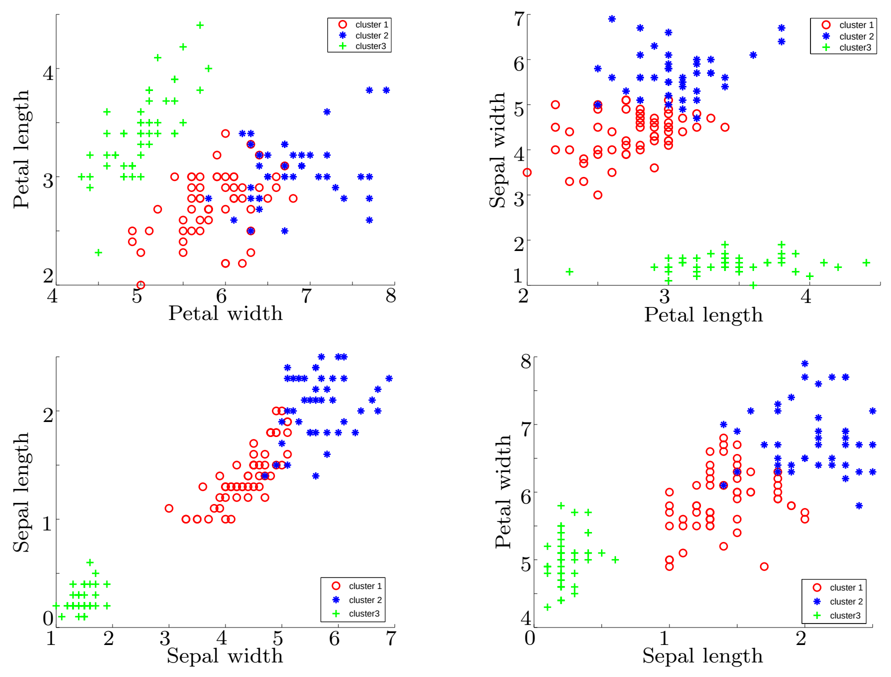


Figure 1.10: *Visualisation of the result of our clustering algorithm. Points that belong to cluster 1 are plotted with* o, *point belonging to cluster 2 are plotted with* $*$, *and points belonging to cluster 3 are plotted with* $+$. *It turns out that cluster 1 corresponds to versicolor flowers, cluster 2 to virginica flowers and cluster 3 to setosa flowers.*

1.1.2.2 Nonnegative matrix factorization without nonnegativity constraints on the factors with application in image compression

The (approximate) nonnegative matrix factorization without nonnegativity constraints on the factors can be stated as

Problem 1.3. *Given $M \in \mathbb{R}_+^{m_1 \times m_2}$ and $a \in \mathbb{N}$, minimize $D_{\mathrm{KL}}(M||VH)$ with respect to V ($\in \mathbb{R}^{m_1 \times a}$) and H ($\in \mathbb{R}^{a \times m_2}$), subject to the constraint $VH \geq 0$.*

In this thesis iterative update formulas to solve this factorization problem will be derived.

As explained before, the need for the nonnegative matrix factorization without nonnegativity constraints on the factors comes from theoretical research concerning hidden Markov models. However, the nonnegative matrix factorization without nonnegativity constraints on the factors has applications that have nothing to do with hidden Markov models. One of these applications is modeling a database of similar images. We here give an introduction to this problem.

In the original paper of Lee and Sueng [72], a database of facial images is modeled using a nonnegative matrix factorization. Each face in the database is represented by a nonnegative column vector, leading to a nonnegative matrix M with the number of columns equal to the number of faces in the database. Next, the matrix is factored as the product VH with the matrices V and H nonnegative. The columns of V contain *eigenfaces* and the columns of H are nonnegative weights that reconstruct the original faces as linear combinations of the eigenfaces. By carrying out an approximation of M as VH with V and H nonnegative, an efficient compression of the original database can be obtained. A first argument for using the nonnegative matrix factorization instead of the SVD or other matrix decompositions is that the reconstructed images VH contain only nonnegative elements and are therefore interpretable as facial images. A second argument is that the eigenfaces are typically sparse (i.e. many zero elements) because of their nonnegativity, which makes that there is a *part-based* interpretation of decomposition, i.e. each face is decomposed into sum of a small number of *parts* (nose, eyes, ears, ...). In our opinion, the first argument is always relevant, while the second is not important in certain applications (for instance in the application where the decomposition is only used for data compression). We therefore consider the decomposition of the nonnegative matrix M into a nonnegative product VH, but where the factors V and H themselves are allowed to contain negative values. In Figure 1.11, we show two faces of the CBCL-database of human faces [1] as well as their compressions using the nonnegative matrix factorization without nonnegativity constraints on the factors (MF), the nonnegative matrix factorization (NMF) and the singular value decomposition (SVD). All approximate factorizations have inner dimension $a = 20$. It is visually clear that the nonnegative matrix factorization without nonnegativity constraints on the factors gives the best results. In the thesis we go into more detail.

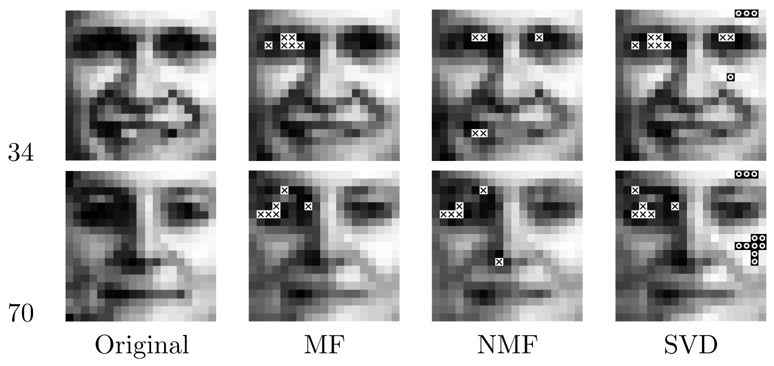


Figure 1.11: *Plot of the i-th face of the database, with* $i = 34, 70$*, and their reconstructions using the nonnegative matrix factorization without nonnegativity constraints on the factors (MF), the nonnegative matrix factorization (NMF) and the singular value decomposition (SVD). All approximate factorizations have inner dimension* $a = 20$*. Pixel values smaller than* 0 *are indicated with* o *and pixel values higher than* 1 *are indicated with* $\times$*.*

1.2 Chapter-by-chapter overview

Figure 1.12 shows the outline of this thesis. Chapter 3 formally introduces hidden Markov models and linear stochastic models. In the next chapters, we explain how to obtain a hidden Markov model from data, either by realization (Chapter 4 and Chapter 5) of by identification (Chapter 6). Both methods make use of matrix factorization techniques (Chapter 2). Once a model is obtained, that model can be used for estimation problems (Chapter 7). A chapter-by-chapter overview of this thesis is now given.

- **Chapter 2** adresses matrix factorizations. We start with a review of the singular value decomposition. We show the low rank SVD approximation of matrices with a certain type of symmetry, has the same type of symmetry. This property has applications in model reduction of linear time-invariant systems. Subsequently, we review the nonnegative matrix factorization problem and introduce two variants to this problem. The first variant is the structured nonnegative matrix factorization problem. We derive iterative update formulas to solve this problem and prove some properties of the update formulas. We apply the structured nonnegative matrix factorization to a clustering problem based on a distance matrix. As a special case of the structured nonnegative matrix factorization, we consider the symmetric nonnegative matrix factorization problem. The second variant of the nonnegative matrix factorization is the nonnegative matrix factorization without nonnegativity constraints on the factors. We provide an algorithm to solve this problem and apply the factorization to the modeling of a database of human faces.

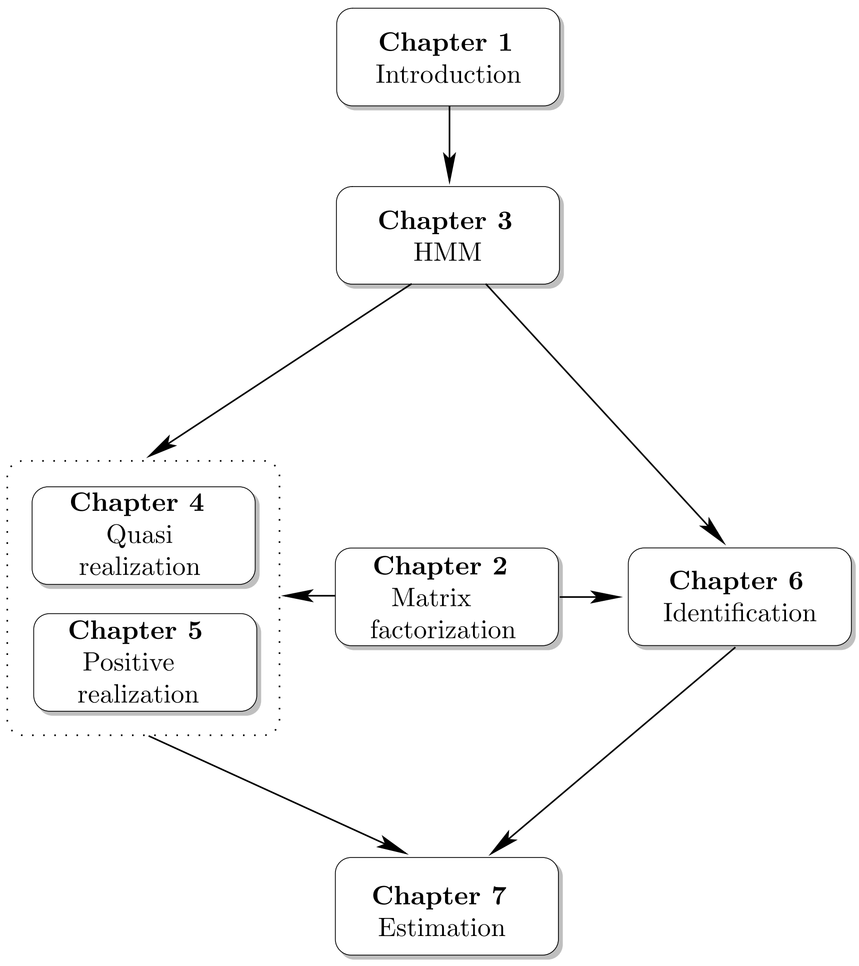


Figure 1.12: *Outline of this thesis.*

Publications related to this chapter: [103, 104, 108, 109].

- **Chapter 3** formally defines hidden Markov models. A distinction is made between Mealy and Moore models and between positive and quasi models. Next, minimality of hidden Markov models is considered and a procedure is proposed to find a minimal quasi Mealy model that is equivalent to a given (nonminimal) model. Subsequently, the equivalence problem for hidden Markov models is considered. In fact, as explained before, the equivalence problem is the third subproblem of the realization

problem. However, it is discussed in this section apart from the realization problem. Concerning the equivalence problem, we make a distinction between Mealy models, quasi Mealy models and Moore models. Finally, linear stochastic models are introduced and the equivalence problem for linear stochastic models is reviewed. It turns out that the solution to the equivalence problem for hidden Markov models is completely analogous to the equivalence problem for linear stochastic models.
Publications related to this chapter: [102, 107].

- **Chapter 4** considers the quasi realization problem for hidden Markov models. We review the exact quasi realization problem, where a model has to be obtained from an infinite amount of exact string probabilities. The partial quasi realization problem is considered, where the string probabilities are still exact but only string probabilities of strings up to a certain length t are given. The approximate quasi realization problem builds a model from approximate string probabilities of strings up to a certain length t. The approximate quasi realization problem is applied to the problem of modeling a DNA sequence. Finally, the realization problem for linear stochastic models is reviewed an it is shown that its solution is analogous to the solution of the quasi realization problem for hidden Markov models.
Publications related to this chapter: [106, 110].

- **Chapter 5** deals with the positive realization problem for hidden Markov models. First, the exact positive realization problem is briefly reviewed. Subsequently, the partial realization problem is reviewed where only string probabilities of string up to length t are to be modeled by the hidden Markov model. A well-known solution in terms of a $t-1$-step Markov model is discussed [114]. Next, we state the approximate partial realization problem and derive iterative update formulas that solve the problem approximately. We make a distinction between the Moore and Mealy realization problem. Simulation examples show the effectiveness of the proposed algorithms. Finally, the approximate realization procedure is applied to the problem of modeling the output sequence of a coin flipping experiment as well as to the problem of modeling DNA sequences.
Publications related to this chapter: [104, 109].

- **Chapter 6** considers the identification problem for hidden Markov models. First the classical Baum-Welch approach is presented. This approach uses the Expectation-Maximization algorithm to solve the problem in Maximum-Likelihood sense. Next, a new method is proposed based on the well-known subspace identification approach for linear stochastic models. The methods first estimates the state sequence from the given output sequence. Subsequently, the system matrices are calculated from the given output sequence and the obtained state sequence. A simulation example shows that the proposed methods works well compared to the classical Baum-Welch approach. The subspace

inspired identification procedure is applied to the modelling of DNA sequences of the HIV genome.
Publications related to this chapter: [105].

- **Chapter 7** deals with different estimation problems for hidden Markov models. Both state as well as output estimation problems are considered. The different estimation problems that are considered are filtering, estimation and (fixed-point, fixed-lag and fixed-interval) smoothing problems. We show that for the output filtering, output estimation and fixed-point and fixed-lag output smoothing problems, it suffices to have a quasi hidden Markov realization instead of a positive hidden Markov realization. This observation is important for two reasons. First of all a quasi realization is easier to obtain than a positive realization. Moreover, a quasi realization typically has lower order than a positive realization which makes the estimation problem less complex. Next, the Viterbi algorithm is reviewed for the fixed interval state smoothing problem and we propose a method for the fixed-interval smoothing problem. The methods of this section are applied to a filtering problem related to the coin flipping experiment of Chapter 5. Finally, a technique for determining the operation mode of a switched hidden Markov model is proposed. This technique is applied to the problem of finding motifs in DNA sequences.
Publications related to this chapter: [106, 110].

1.3 Personal contributions

This section summarizes the personal contributions of this thesis.

1.3.1 Matrix factorizations

In this section we describe our contributions concerning matrix factorizations.

- Given a matix M that obeys $M = PMQ$ where P and Q are unitary matrices, we prove that the rank-k SVD truncation of the matrix M has the same type of symmetry as M (Section 2.1.3). This theorem allows to prove for instance that the rank-k SVD-truncation of a circulant matrix is again circulant.
Publications related to this topic: [103].

- We introduce the structured nonnegative matrix factorization problem (Section 2.3.1): calculate an approximation $VAV^\top$ to a given matrix P that is optimal with respect to the Kullback-Leibler divergence between P and $VAV^\top$. We first show that it holds for a stationary point of the divergence that the element sum of P is equal to the element sum of the approximation. We propose iterative formulas of which we prove that, if they converge, they converge to a stationary point of the divergence. The structured nonnegative matrix factorization problem is succesfully applied

to the clustering of data points based on their distance matrix (Section 2.3.2). Finally, the symmetric nonnegative matrix factorization problem is introduced and solved (Section 2.4).
Publications related to this topic: [104, 109].

- We introduce the nonnegative matrix factorization problem without nonnegativity constraints on the factors (Section 2.5). We propose iterative formulas to solve this matrix factorization problem. Finally, the nonnegative matrix factorization without nonnegativity constraints is applied to the modeling of a database of facial images (Section 2.5.4).
Publications related to this topic: [108].

1.3.2 Realization of hidden Markov models

We contributed to the realization problem (both quasi and positive) and the equivalence problem for hidden Markov models. We here present an overview.

- We develop a test for checking whether a given quasi Mealy model is minimal and a procedure to find a minimal quasi Mealy model that is equivalent to a given nonminimal Mealy model (Section 3.2.4). We provide a test to check whether two positive Mealy hidden Markov models are equivalent and give a description of the complete set of equivalent models (Section 3.3.1.2). We prove that Moore models that are minimal as a quasi Mealy model, under certain conditions, do not have non-trivial equivalents. Moore models that are not minimal as a quasi model can have equivalents. We provide a test for checking the equivalence of Moore models and describe the complete set of equivalent models (Section 3.3.2).
Publications related to this topic: [102, 107].

- We introduce the partial quasi realization problem. Given partial string probabilities, we show that a quasi HMM that realizes these string probabilities can always be found (Section 4.2.1). However, the minimal partial realization problem is hard to solve. We introduce and solve the pseudo realization problem, a relaxed version of the partial quasi realization problem (Section 4.2.2). Next, we consider the approximate pseudo realization problem and provide methods to solve it (Section 4.3). The first methods make a low rank approximation of the generalized Hankel matrix of string probabilities and subsequently apply the realization algorithm. The last method builds a full order balanced model and subsequently reduces this model to obtain an approximate quasi realization. The approximate quasi realization algorithms are applied to the problem of modeling DNA sequences (Section 4.4).
Publications related to this topic: [106, 110].

- We consider the approximate partial realization problem for Moore and for Mealy hidden Markov models where string probabilities are given for strings up to length t. We show that the structured nonnegative matrix

factorization (Section 5.3.1) can be used to solve the Moore realization problem for $t = 2$. Further, we propose a solution to the Mealy realization problem for string probabilities of strings up to length t (Section 5.3.2). We perform simulation examples to show the quality of the proposed methods (Section 5.4). The approximate realization algorithms are applied to the problems of modeling the output sequence of a coin flipping experiment (Section 5.5) and the problem of modeling DNA sequences (Section 5.6).
Publications related to this topic: [104, 109].

1.3.3 Identification of hidden Markov models

We now describe our contributions to the identification problem for hidden Markov models.

- We derive the Baum-Welch algorithm for Mealy hidden Markov models. To the best of our knowledge Baum-Welch had only been considered for Moore models.
- We propose a new method for the identification of hidden Markov models (Section 6.3). The method first estimates string probabilities of strings up to a certain length. These string probabilities are stacked in a matrix and the nonnegative matrix factorization applied on this matrix yields a way to estimate the state sequence corresponding to the given output sequence. Next, the system matrices can be calculated from the obtained state sequence and the given output sequence by solving a least squares problem. In a simulation example (Section 6.4), it is shown that the proposed method outperforms the existing Baum-Welch identification method. The subspace inspired method is used to model DNA sequences of the HIV genome.
  Publications related to this topic: [105].

1.3.4 Estimation for hidden Markov models

Finally, we describe our contributions to the estimation problem for hidden Markov models.

- We derive formulas for the recursive filtering, prediction and fixed-point and fixed-lag smoothing problem (Section 7.2 and Section 7.3). We show that for the recursive output filtering, output prediction and fixed-point and fixed-lag output smoothing problem, it suffices to have a quasi hidden Markov realization instead of a positive realization. We introduce and solve the fixed-interval output smoothing problem (Section 7.4.2). We apply the filtering techniques of this chapter to a problem concerning the coin flipping experiment (Section 7.5). Finally, we propose a technique for determining the operation mode of a switched HMM (Section 7.6) and apply this technique to the problem of finding motifs in DNA sequences (Section 7.7).
  Publications related to this topic: [106, 110].

Chapter 2

Matrix factorizations

Recent technological developments in sensor technology and computer hardware have resulted in increasing quantities of data. Processing these large amounts of data has created new concerns with respect to data representation and dimensionality reduction. An important problem in dimensionality reduction is the low rank matrix approximation problem.

The general low rank matrix approximation problem of an $m_1 \times m_2$ real matrix M with a product VH with inner dimension a ($a < \min\{m_1, m_2\}$) consists of minimizing $D(M, VH)$ over V and H, where $D(X, Y)$ is a distance measure between X and Y. If there are no additional constraints on the factors V and H, the singular value decomposition gives a decomposition that is optimal in the Frobenius distance. For an early review in the singular value decomposition, we refer to [95]. There exist efficient algorithms to compute the singular value decomposition [55].

Often the data to be analyzed are nonnegative, and the low rank data are required to be nonnegative too. Classical methods can not guarantee to maintain the nonnegativity. Imposing a nonnegativity constraint on the factors of the decomposition, however, makes the matrix factorization problem non convex and difficult to solve. The exact nonnegative matrix factorization problem was stated in [85, 100]. The minimal inner dimension of an exact decomposition is called the positive rank. It is shown in [28] that there exists a finite algorithm to compute the positive rank of a given matrix. However, the complexity bounds on the number of arithmetic/boolean operations that this algorithm requires, are non-polynomial. In [111], it is shown that the computation of the positive rank is NP-hard. The approximate nonnegative matrix factorization problem, where one looks for a local minimum of a particular cost function, has been first introduced by Lee and Sueng [72]. Lee and Sueng also propose multiplicative update formulas to solve the approximate factorization problem. The convergence to a fixed point of the cost function is proven in [48, 73].

Since the introduction of the approximate nonnegative matrix factorization problem, it received lots of interest, both theoretically, algorithmically as well as in practical applications. Next to the multiplicative update algorithms, two other classes of algorithms have been considered in the literature: gradient descent methods [92] and alternating least squares methods [79]. It has been proven in [57] that the row and column sum of the approximation is equal to the row and column sum of the original matrix when the Kullback-Leibler divergence is used as cost function. In [21] weighted nonnegative matrix factorization algorithms are presented that allow to emphasize parts of the data matrix to be approximated better than orther parts. In [25] the nonnegative matrix factorization for symmetric matrices has been considered. The approximate nonnegative matrix factorization has been applied in image compression, clustering, data analysis [16, 40, 74, 80].

Recently, extensions to the nonnegative matrix factorization problem have been introduced. In [117] the nonnegative tensor factorization is considered and in [41] tri-factorizations, factorizations in three factors are considered.

In this chapter we introduce two modifications to the approximate nonnegative matrix factorization. First of all, the structured nonnegative matrix factorization is introduced. Here, a square matrix P has to be decomposed as a product $VAV^\top$ with V and A elementwise nonnegative. The second modification is the nonnegative matrix factorization without nonnegativity constraints on the factors. In that problem the matrix M has to be decomposed into a product VH where VH is elementwise nonnegative, but where both V and H are allowed to contain negative values.

List of own contributions

We here describe our contributions to matrix approximation problems.

- In Section 2.1.3 we consider matrices M that obey the symmetry: $M = PMQ$ where P and Q are unitary matrices. We show that the optimal rank k approximation M_k calculated using the singular value decomposition of this type of symmetric matrices, has the same type of symmetry, i.e. $M_k = PM_kQ$. We show that circulant matrices are one example of matrices with this type of symmetry.

- In Section 2.3 we introduce the structured nonnegative matrix factorization: approximate a square nonnegative matrix P by a product $VAV^\top$ where V and A are elementwise nonnegative. We propose multiplicative update formulas to solve this problem approximately in the Kullback-Leibler divergence and prove that if the update formulas converge that they converge to a fixed point of the divergence between P and its approximation $VAV^\top$. We apply the proposed technique to the problem of clustering points based on their distance matrix.

- In Section 2.4 we introduce the symmetric nonnegative matrix factorization problem: approximate a square nonnegative matrix P with a product $VV^\top$ where V is elementwise nonnegative. This problem can be considered as a special case of the structured nonnegative matrix factorization problem. We again propose multiplicative update formulas and prove their convergence.

- In Section 2.5 we introduce the nonnegative matrix factorization without nonnegativity constraints on the factors. We explain the importance of this problem for applications and derive update formulas to solve the problem approximately. Subsequently, we note that it is surprising that much attention has been paid to the nonnegativity constraint on the elements of an approximation, which in essence is a lower bound constraint, while in many applications upper bounds are as important as lower bounds. The nonnegative matrix factorization without nonnegativity constraints on the factors allows to deal with upper as well as lower bounds on the elements of the approximation. Finally, we apply the decomposition to the problem of compressing a database of facial images.

Section-by-section overview

In Section 2.1 we review the singular value decomposition and in Section 2.2 we review the classical nonnegative matrix factorization problem. In Section 2.3 the structured nonnegative matrix factorization is introduced and solved approximately. In Section 2.4 we consider the symmetric nonnegative matrix factorization. In Section 2.5 finally, the nonnegative matrix factorization problem without nonnegativity constraints on the factors is introduced.

2.1 Singular value decomposition

In this section we review the Singular Value Decomposition (SVD). In Section 2.1.1 we introduce the full and reduced singular value decomposition. In Section 2.1.2 we explain how the SVD can be used for optimal low rank matrix factorization problems. In Section 2.1.3 we show that the low rank SVD approximation of a matrix with a certain type of symmetry, possesses the same type of symmetry.

2.1.1 Main theorem

A square matrix $P \in \mathbb{R}^{n\times n}$ is said to be *unitary* if $P^\top P = I = PP^\top$. Using this definition, we are able to state the main theorem concerning the singular value decomposition of a matrix $M \in \mathbb{R}^{m_1\times m_2}$.

Theorem 2.1. *Given a matrix $M \in \mathbb{R}^{m_1\times m_2}$ of rank r. Then there exists a decomposition given by*

$$M = U\Sigma V^\top$$

where $U \in \mathbb{R}^{m_1\times m_1}$ and $V \in \mathbb{R}^{m_2\times m_2}$ are unitary and

$$\Sigma = \begin{bmatrix} \Sigma_{(1)} & 0 \\ 0 & 0 \end{bmatrix}$$

with

$$\Sigma_{(1)} := \operatorname{diag}(\sigma_1(M), \sigma_2(M), \ldots, \sigma_r(M)),$$

with

$$\sigma_1(M) \geq \sigma_2(M) \geq \ldots \geq \sigma_r(M) > 0.$$

This decomposition is called the Singular Value Decomposition (SVD) *of M.*

The elements $\sigma_1(M), \sigma_2(M), \ldots, \sigma_r(M)$ are called the *singular values* of M, the columns $U_{:,1}$, $U_{:,2}$, ..., $U_{:,m_1}$ of U are called the *left singular vectors* of M and the columns $V_{:,1}$, $V_{:,2}$, ..., $V_{:,m_1}$ of V the *right singular vectors* of M.

From the SVD of M, we obtain its dyadic decomposition

$$M = \sum_{i=1}^{r} \sigma_i(M) U_{:,i} V_{:,i}^\top$$

where $U_{:,i}$ and $V_{:,i}$ are the i-th columns of U and V respectively, which correspond to the i-th singular value $\sigma_i(M)$.

It is clear from Theorem 2.1 that the matrices involved in the SVD can be reduced. The last $m_1 - r$ columns of U and the last $m_2 - r$ rows of $V^\top$ are multiplied with zeros from Σ. By leaving out these columns and rows, we obtain the *reduced SVD*.

Theorem 2.2. *Given a matrix $M \in \mathbb{R}^{m_1 \times m_2}$ of rank r. Then it can be decomposed as*

$$M = U_{(1)} \Sigma_{(1)} V_{(1)}^\top,$$

where the columns of $U_{(1)} \in \mathbb{R}^{m_1 \times r}$ and $V_{(1)} \in \mathbb{R}^{m_2 \times r}$ are orthonormal, i.e. $U_{(1)}^\top U_{(1)} = V_{(1)}^\top V_{(1)} = I_r$ and $\Sigma_{(1)}$ is defined as in Theorem 2.1. This decomposition is called the reduced Singular Value Decomposition (reduced SVD) *of M.*

2.1.2 Optimal rank k approximation

Define the *rank k SVD-truncation* of M with $k \leq r$ as

$$M_k := U_{(1)} \begin{bmatrix} \Sigma_k & 0 \\ 0 & 0 \end{bmatrix} V_{(1)}^\top$$

with $\Sigma_k := \operatorname{diag}(\sigma_1(M), \sigma_2(M), \ldots, \sigma_k(M))$. It is well-known that, if the *gap condition*

$$\sigma_k(M) > \sigma_{k+1}(M)$$

holds, then the rank k SVD-truncation of M is uniquely defined. Indeed, while the $\sigma(M)$'s are always uniquely defined, U and V are never unique, but nevertheless, if the gap condition holds, then the rank k SVD-truncation of M is unique.

The norm $||\cdot||$ on $\mathbb{R}^{m_1 \times m_2}$ is said to be *unitarily invariant* if $(M \in \mathbb{R}^{m_1 \times m_2}) \wedge (P, Q \text{ unitary}) \Rightarrow ||PMQ|| = ||M||$. An example of a unitarily invariant norm is the Frobenius norm. The *Frobenius norm* of $M = [M_{ij}] \in \mathbb{R}^{m_1 \times m_2}$ is defined as $||M||_F := \sqrt{\sum_{i=1}^{m_1} \sum_{j=1}^{m_2} (M_{ij})^2}$. Now, the following theorems are well-known.

Theorem 2.3. *The* rank k *SVD-truncation of M leads to an optimal rank k approximation of M with respect to any unitarily invariant norm. In other words*

$$(||\cdot|| \textit{ unitarily invariant}) \wedge (\operatorname{rank} M' \leq k) \Rightarrow ||M - M'|| \geq ||M - M_k||.$$

Theorem 2.4. *If the gap condition $\sigma_k(M) > \sigma_{k+1}(M)$ holds, then the* rank k *SVD-truncation M_k is the* unique *matrix of rank k which approximates M optimally in the Frobenius norm, i.e.*

$$\big(\sigma_k(M) > \sigma_{k+1}(M)\big) \wedge (\operatorname{rank} M'_k \leq k) \\ \wedge (||M - M'_k||_F = ||M - M_k||_F) \Rightarrow M'_k = M_k$$

Proof: This theorem is well-known [44], but for the sake of completeness, we give a proof in Appendix A. ■

Of course, it follows that if the gap condition $\sigma_k(M) > \sigma_{k+1}(M)$ holds, then the rank k SVD-truncation M_k is the *unique* matrix of rank k which approximates M optimally, simultaneously for all unitarily invariant norms. It is an interesting question to check for which unitarily invariant norms the analogue of Theorem 2.4 holds.

2.1.3 Optimal rank k approximation of matrices with symmetry

Using Theorem 2.4, we are able to prove the following theorem about the SVD of a matrix with a certain type of symmetry.

Theorem 2.5. *Assume that the matrix $M \in \mathbb{R}^{m_1 \times m_2}$ has the following* symmetry

$$M = PMQ$$

with P and Q unitary matrices. Then, if $\sigma_k(M) > \sigma_{k+1}(M)$, M_k, the rank k approximation derived from truncating the SVD, has the same symmetry

$$M_k = PM_kQ.$$

Proof: The Frobenius norm is unitarily invariant, so

$$||M - M_k||_F = ||P(M - M_k)Q||_F = ||M - PM_kQ||_F.$$

Hence PM_kQ is an optimal rank k approximation of M with respect to the Frobenius norm. By the uniqueness shown in Theorem 2.4, $PM_kQ = M_k$. ■

We now provide some examples of matrices $M \in \mathbb{R}^{m_1 \times m_2}$ for which $M = PMQ$ with P and Q unitary matrices.

Matrices with equal rows/columns Let $P_{i,j}$ be the $m_1 \times m_1$ permutation matrix such that in $P_{i,j}x$ the i-th and j-th elements of x are permuted. Then in $P_{i,j}M$ the i-th and j-th rows are permuted. Now $M = P_{i,j}M$ means that the i-th and the j-th rows of M are equal. Theorem 2.5 allows us to conclude that if the gap condition holds, then $M_k = P_{i,j}M_k$, i.e. the i-th and j-th rows of M_k are also equal. A matrix M for which the symmetry $M = P_{i,j}M$ holds for many pairs of (i, j), corresponds to either a matrix with more than two equal rows or a matrix with more than one group of rows which are identical. If the gap condition holds, all these symmetries separately are retained after SVD-truncation. Analogous results can be obtained for the columns of M.

Matrices with zero-rows/-columns To express that the i-th row of M is zero, consider the matrix $P_i = \mathrm{diag}(1, \ldots, 1, -1, 1, \ldots, 1)$, with the -1 on the i-th position, and express that $M = P_iM$. If the gap condition holds, then for the optimal rank k approximation of M holds that $M_k = P_iM_k$, i.e. the i-th row of M_k is also equal to zero. If the symmetry $M = P_iM$ holds for different values of i, then more than one row of M is equal to zero. All the symmetries separately are retained after SVD-truncation if the gap condition holds. Analogous results can be obtained for the columns of M.

Example 2.1. *In this example we consider a matrix M with three differents symmetries of the form $M = PMQ$ where P and Q are unitary matrices. Using Theorem 2.5, we show that the SVD-truncations of M have the same type of symmetries.*

The matrix M is given by

$$M = \begin{bmatrix} 4 & 2 & 5 & 9 & 2 \\ 1 & 2 & 3 & 5 & 2 \\ 4 & 2 & 5 & 9 & 2 \\ 2 & 6 & 4 & 7 & 6 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The three symmetries are the following. First of all, the first and the third row of M are equal to each other. Second, the fifth row of M is a zero-row. Finally, the second and the fifth column of M are equal. The reduced SVD of M is given by

$$M = \begin{bmatrix} 0.55 & -0.42 & 0.16 \\ 0.32 & 0.05 & -0.95 \\ 0.55 & -0.42 & 0.16 \\ 0.54 & 0.80 & 0.23 \\ 0.00 & 0.00 & 0.00 \end{bmatrix} \begin{bmatrix} 20.39 & & \\ & 5.23 & \\ & & 0.91 \end{bmatrix} \begin{bmatrix} 0.28 & 0.30 & 0.42 & 0.75 & 0.30 \\ -0.32 & 0.63 & -0.15 & -0.30 & 0.63 \\ 0.89 & 0.13 & -0.34 & -0.25 & 0.13 \end{bmatrix}.$$

From $\sigma_1(M) > \sigma_2(M)) > \sigma_3(M)$ and Theorem 2.5, it follows that the low rank SVD-truncations of M have the same three symmetries. Indeed, the rank 2 and the rank 1 SVD-truncations of M are given by

$$M_2 = \begin{bmatrix} 3.87 & 1.98 & 5.05 & 9.04 & 1.98 \\ 1.76 & 2.11 & 2.71 & 4.79 & 2.11 \\ 3.87 & 1.98 & 5.05 & 9.04 & 1.98 \\ 1.82 & 5.97 & 4.07 & 7.05 & 5.97 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix},$$

and

$$M_1 = \begin{bmatrix} 3.18 & 3.35 & 4.73 & 8.39 & 3.35 \\ 1.85 & 1.94 & 2.75 & 4.87 & 1.94 \\ 3.18 & 3.35 & 4.73 & 8.39 & 3.35 \\ 3.15 & 3.32 & 4.69 & 8.31 & 3.32 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix}.$$

□

Circulant matrices In this section we consider block matrices with $n \times n$ blocks of size $p \times m$. Define the special permutation matrix $\Pi \in \mathbb{R}^{n \times n}$

$$\Pi = \begin{bmatrix} 0 & I_{n-1} \\ 1 & 0 \end{bmatrix}.$$

Let $F = [\; F_1^\top \;\; \dots \;\; F_n^\top \;]^\top$ with $F_i \in \mathbb{R}^{p \times m}$, $i = 1, \dots, n$, then the block matrix $\mathcal{C}_F$ with $n \times n$ blocks of size $p \times m$

$$\mathcal{C}_F := \begin{bmatrix} F & (\Pi \otimes I_p)F & (\Pi \otimes I_p)^2 F & \cdots & (\Pi \otimes I_p)^{n-1} F \end{bmatrix}, \tag{2.1}$$

is called the *block circulant matrix generated by* F [30]. Such a matrix looks like

$$\mathcal{C}_F = \begin{bmatrix} F_1 & F_2 & \dots & F_{n-1} & F_n \\ F_2 & F_3 & \dots & F_n & F_1 \\ \vdots & \vdots & & \vdots & \vdots \\ F_{n-1} & F_n & \dots & F_{n-3} & F_{n-2} \\ F_n & F_1 & \dots & F_{n-2} & F_{n-1} \end{bmatrix}.$$

Observe the *block Hankel* structure of block circulant matrices. An equivalent way of defining block circulant matrices is:

$$[M \in \mathbb{R}^{mp \times mm} \text{ is block circulant}] \iff [M = (\Pi \otimes I_p) M (\Pi \otimes I_m)].$$

It follows from Theorem 2.5 that if M is block circulant and if the gap condition holds, then the truncated SVD M_k is also block circulant. This observation has applications in the domain of model reduction of linear time-invariant systems with symmetries. It allows to prove that the finite-time balanced reduced model of a given periodic system is periodic with the same period as the full-order system [96,103]. It also allows to prove that the balanced reduced model of a system with pointwise symmetries in the impulse response has the same symmetries in its impulse response. In [103] we prove this Theorem and provide an example of a system with pointwise symmetric impulse response, inspired by the work of [101].

Example 2.2. *In this example we calculate the low rank SVD-truncation of a circulant matrix M. We show that, if the gap condition holds, the low rank SVD-truncation of M is again circulant (using Theorem 2.5).*

The matrix M is given by

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \\ 3 & 4 & 5 & 1 & 2 \\ 4 & 5 & 1 & 2 & 3 \\ 5 & 1 & 2 & 3 & 4 \end{bmatrix}.$$

Note that the blocksize of M is 1×1. The singular values of M are 15, 4.25, 4.25, 2.63 and 2.63. It follows from Theorem 2.5 that the rank 3 and rank 1 approximations of M are circulant. Indeed, M_3 and M_1 are given by

$$M_3 = \begin{bmatrix} 2.00 & 1.38 & 3.00 & 4.62 & 4.00 \\ 1.38 & 3.00 & 4.62 & 4.00 & 2.00 \\ 3.00 & 4.62 & 4.00 & 2.00 & 1.38 \\ 4.62 & 4.00 & 2.00 & 1.38 & 3.00 \\ 4.00 & 2.00 & 1.38 & 3.00 & 4.62 \end{bmatrix} \tag{2.2}$$

and

$$M_1 = \begin{bmatrix} 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 \end{bmatrix}.$$

□

We know from Theorem 2.4 that if the gap condition holds, the rank k SVD-truncation M_k is the unique matrix of rank k which approximates M optimally in the Frobenius norm. As a consequence of this, the SVD-truncation M_k of a block circulant matrix can very nicely be computed using the Discrete Fourier Transform (DFT). We explain this only for the vector case. Consider

$$M = \begin{bmatrix} m_1 & m_2 & \dots & m_{n-1} & m_n \\ m_2 & m_3 & \dots & m_n & m_1 \\ \vdots & \vdots & & \vdots & \vdots \\ m_{n-1} & m_n & \dots & m_{n-3} & m_{n-2} \\ m_n & m_1 & \dots & m_{n-2} & m_{n-1} \end{bmatrix},$$

with $m_t \in \mathbb{R}^p$ for $t = 1, 2, \dots n$. Let

$$\tilde{m}_f := \sum_{t=1}^{n} m_t e^{-if\frac{2\pi}{n}t}, \quad f = 0, 1, \dots, n-1$$

be the DFT of the first block row of M: $m_1, m_2, \dots, m_n$, such that

$$m_t = \frac{1}{n} \sum_{f=0}^{n-1} \tilde{m}_f e^{if\frac{2\pi}{n}t}, \quad t = 1, 2, \dots, n.$$

It follows that the rank of M equals the cardinality of the set

$$\{f \in \{0, 1, \cdots, m-1\} \mid ||\tilde{m}_f|| \neq 0\}.$$

It is also known that

$$\frac{1}{n}||M||_F^2 = \sum_{t=1}^{n} ||m_t||^2 = \frac{1}{n}\sum_{f=0}^{n-1} ||\tilde{m}_f||^2.$$

Therefore, in order to obtain M_k, an optimal rank k approximation of M in the Frobenius norm, we can proceed as follows. First calculate

$$\hat{m}_t = \frac{1}{n}\sum_{f \in F_k} \tilde{m}_f e^{if\frac{2\pi}{n}t}, \quad t = 1, 2, \ldots, n,$$

with F_k the subset of $\{0, 1, \ldots, n-1\}$ of cardinality k with the property

$$[(f \in F_k) \wedge (f' \notin F_k)] \Rightarrow [||\tilde{m}_f|| \ \geq || \ \tilde{m}_{f'}||].$$

Now, it is easy to see that M_k is equal to the block circulant matrix induced by the vector $\begin{bmatrix} \hat{m}_1^\top & \hat{m}_2^\top & \ldots & \hat{m}_n^\top \end{bmatrix}^\top$ (see Equation (2.1)). If for each $f \in F_k$ there exists a $f' \in F_k$ such that $\tilde{m}_{f'}$ is the complex conjugate of $\tilde{m}_f$, then M_k is real. Note also that M_k is the unique optimal rank k approximation of M in the Frobenius norm if

$$[(f \in F_k) \wedge (f' \notin F_k)] \Rightarrow [||\tilde{m}_f|| \ > \ ||\tilde{m}_{f'}||].$$

Assume that both these conditions are satisfied. Then M_k approximates M optimally in the Frobenius norm with a block circulant matrix of rank k and it is the unique block circulant matrix that does so. Hence we derived an alternative way to calculate the SVD-truncation M_k by making use of the DFT. Moreover, since $\hat{m}_t$, $t = 1, 2, \ldots n-1$ may be computed using the Fast Fourier Transform (FFT), it is numerically much more efficient to compute M_k by first computing $\hat{m}_t$, $t = 1, 2, \ldots n-1$ and then forming M_k, than it is to compute the SVD. This observation is valid also when we look for an optimal rank k approximation of M in another unitarily invariant norm than the Frobenius norm.

Example 2.3. *We continue with the matrix M of Example 2.2 and show that the low rank SVD-truncation of a circulant matrix can be obtained using the DFT. We here consider the rank 3 case. Calculating the DFT of the first row of M, truncating the smallest DFT values and subsequently calculating the inverse discrete Fourier transform yields*

$$\begin{array}{c} [1,\ 2,\ 3,\ 4,\ 5] \\ \downarrow \quad \text{DFT} \\ [15,\ -2.50+3.44i,\ -2.50+0.81i,\ -2.50-0.81i,\ -2.50-3.44i] \end{array}$$

$$\begin{gathered}\downarrow\ \text{truncating}\\ [15,\ -2.50+3.44i,\ 0,\ 0,\ -2.50-3.44i]\\ \downarrow\ \text{inverse DFT}\\ [2,\ 1.38,\ 3,\ 4.61,\ 4].\end{gathered}$$

So, using the discrete Fourier transform, we find back the same rank 3 approximation of M as was obtained by truncating the SVD (Equation (2.2)). □

2.2 Nonnegative matrix factorization

In this section we briefly review the classical nonnegative matrix factorization problem as it forms the basis of several new nonnegative matrix factorizations algorithms.

The *exact nonnegative matrix factorization problem* can be stated as follows: given a matrix $M \in \mathbb{R}_+^{m_1 \times m_2}$, find a decomposition $M = VH$ with $V \in \mathbb{R}_+^{m_1 \times a}$ and $H \in \mathbb{R}_+^{a \times m_2}$, and with a as small as possible. The minimal inner dimension a for which a decomposition exists is called the *positive rank* (p–rank) of M. There exist matrices with only trivial minimal decompositions $M = DH$ and $M = VD$, where D is a nonnegative diagonal matrix. In [33] these matrices are called prime. From the definitions of the rank and the positive rank of a matrix, it follows that $0 \leq \operatorname{rank} M \leq \text{p–rank}\, M \leq \min\{m_1, m_2\}$. It is shown in [28] that there exists a finite algorithm to compute the positive rank of a given matrix. However, the complexity bounds on the number of arithmetic/boolean operations that this algorithm requires, are non-polynomial. Moreover, in [111], it is shown that the computation of the positive rank is NP-hard. Recently, the (approximate) nonnegative matrix factorization problem was introduced in [72]. The idea is that one chooses the inner dimension a and looks for nonnegative matrices V and H such that VH approximates M optimally in a certain distance measure. The Kullback-Leibler divergence is a popular distance measure. The Kullback-Leibler divergence between two nonnegative matrices of the same size is defined as

$$D_{KL}(A||B) := \sum_{ij} (A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij}).$$

The *(approximate) Nonnegative Matrix Factorization (NMF) problem* can now be stated as

Problem 2.1. *Given $M \in \mathbb{R}_+^{m_1 \times m_2}$ and given a, minimize $D_{KL}(M||VH)$ with respect to V (of size $m_1 \times a$) and H (of size $a \times m_2$), subject to the constraints $V \geq 0$, $H \geq 0$.*

In [72,73], Lee and Sueng propose iterative update formulas to solve Problem 2.1 and prove interesting properties of the formulas.

Theorem 2.6. *[72, 73] Given random starting values $V^{(0)} \in \mathbb{R}_+^{m_1 \times a}$ and $H^{(0)} \in \mathbb{R}_+^{a \times m_2}$, then the divergence $D_{KL}(M||VH)$ is nonincreasing under the update rules*

$$H_{il}^{(t+1)} = H_{il}^{(t)} \frac{\sum_\mu V_{\mu i}^{(t)} \frac{M_{\mu l}}{(V^{(t)}H^{(t)})_{\mu l}}}{\sum_\mu V_{\mu i}^{(t)}}, \quad V_{ki}^{(t+1)} = V_{ki}^{(t)} \frac{\sum_\nu H_{i\nu}^{(t)} \frac{M_{k\nu}}{(V^{(t)}H^{(t)})_{k\nu}}}{\sum_\nu H_{i\nu}^{(t)}}$$

The divergence is invariant under these updates if and only if V and H are in a stationary point of the divergence.

The theorem states that fixed points of the update formulas are stationary points of the cost function $D_{KL}(M||VH)$. However, the theorem does not imply convergence of the update formulas. By chosing nonnegative starting values for V and H, it is garantueed that the obtained matrices V and H are nonnegative.

2.3 Structured nonnegative matrix factorization

In this section we introduce the structured nonnegative matrix factorization problem. This matrix factorization is important as such, but is especially of use to solve the approximate positive realization problem for hidden Markov models (Section 5.3.1). In Section 2.3.1 we describe a solution to the structured nonnegative matrix factorization problem and in Section 2.3.2 we apply the decomposition to the clustering of data points based on their distance matrix.

2.3.1 Structured nonnegative matrix factorization

The *exact structured nonnegative matrix factorization problem* may be stated as follows: given a square matrix $P \in \mathbb{R}_+^{p \times p}$, find a decomposition $P = VAV^\top$ with $V \in \mathbb{R}_+^{p \times a}$, $A \in \mathbb{R}_+^{a \times a}$, and with a as small as possible. We define the *structured positive rank* (sp–rank) of a square matrix P as the minimal dimension a for which a decomposition $P = VAV^\top$ exists. From the definition of the different ranks, it is clear that $0 \leq \operatorname{rank} P \leq \text{p–rank}\, P \leq \text{sp–rank}\, P \leq p$. In a similar way as in [28], it can be shown that there exists a finite algorithm to compute the structured positive rank of a given matrix. However, the complexity bounds on the number of arithmetic/boolean operations that this algorithm requires, are non-polynomial. The *(approximate) structured nonnegative matrix factorization problem* is now stated as follows.

Problem 2.2. *Given $P \in \mathbb{R}_+^{p \times p}$ and given a, minimize $D_{KL}(P||VAV^\top)$ with respect to V (of size $p \times a$) and A (of size $a \times a$), subject to the constraints $V \geq 0$, $H \geq 0$.*

The partial derivatives of $F(A, V) = D_{KL}(P||VAV^\top)$ with respect to the elements A_{ij} and V_{ki} of the matrices A and V can be calculated as

$$\frac{\partial F}{\partial A_{ij}}(A, V) = -\sum_{\mu\nu} V_{\mu i} V_{\nu j} \frac{P_{\mu\nu}}{(VAV^\top)_{\mu\nu}} + \sum_{\mu\nu} V_{\mu i} V_{\nu j}, \tag{2.3}$$

$$\begin{aligned}\frac{\partial F}{\partial V_{ki}}(A,V) &= -\sum_{\lambda\nu} V_{\nu\lambda}A_{i\lambda}\frac{P_{k\nu}}{(VAV^\top)_{k\nu}} - \sum_{\lambda\nu} V_{\nu\lambda}A_{\lambda i}\frac{P_{\nu k}}{(VAV^\top)_{\nu k}} \\ &\quad + \sum_{\lambda\nu} V_{\nu\lambda}A_{i\lambda} + \sum_{\lambda\nu} V_{\nu\lambda}A_{\lambda i}. \end{aligned} \tag{2.4}$$

The Karush-Kuhn-Tucker optimality conditions [65, 69] are

$$V_{ki} \ge 0, \qquad A_{ij} \ge 0, \tag{2.5}$$

$$\frac{\partial F}{\partial A_{ij}}(A,V) \ge 0, \qquad \frac{\partial F}{\partial V_{ki}}(A,V) \ge 0, \tag{2.6}$$

$$A_{ij}\frac{\partial F}{\partial A_{ij}}(A,V) = 0, \qquad V_{ki}\frac{\partial F}{\partial V_{ki}}(A,V) = 0, \tag{2.7}$$

for $i = 1, 2, \ldots, a$, $j = 1, 2, \ldots, a$, and $k = 1, 2, \ldots, p$.

Now we prove the following theorem.

Theorem 2.7. *Let $P \in \mathbb{R}^{p\times p}$. Every stationary point (A,V) of the cost function $D_{KL}(P||VAV^\top)$ preserves the mean of the row and column sum of P, i.e.*

$$\frac{\sum_l P_{kl} + P_{lk}}{2} = \frac{\sum_l (VAV^\top)_{kl} + (VAV^\top)_{lk}}{2}, \quad k = 1, 2, \ldots p. \tag{2.8}$$

As a consequence the element sum of P is preserved, i.e.

$$\sum_{kl} P_{kl} = \sum_{kl} (VAV^\top)_{kl}. \tag{2.9}$$

Proof: Equation (2.7) gives, for $k = 1, 2, \ldots, p$ and $i = 1, 2, \ldots, a$,

$$V_{ki}\frac{\partial F}{\partial V_{ki}}(A,V) = 0.$$

From Equation (2.4), we obtain

$$V_{ki}\left(\sum_{\lambda\nu} V_{\nu\lambda}A_{i\lambda}\frac{P_{k\nu}}{(VAV^\top)_{k\nu}} + V_{\nu\lambda}A_{\lambda i}\frac{P_{\nu k}}{(VAV^\top)_{\nu k}}\right) = V_{ki}\left(\sum_{\lambda\nu} V_{\nu\lambda}A_{i\lambda} + V_{\nu\lambda}A_{\lambda i}\right). \tag{2.10}$$

The sum over i of the left hand side of Equation (2.10) gives

$$\begin{aligned}&\sum_i V_{ki}\left(\sum_{\lambda\nu} V_{\nu\lambda}A_{i\lambda}\frac{P_{k\nu}}{(VAV^\top)_{k\nu}} + V_{\nu\lambda}A_{\lambda i}\frac{P_{\nu k}}{(VAV^\top)_{\nu k}}\right) = \\ &\sum_\nu (VAV^\top)_{k\nu}\frac{P_{k\nu}}{(VAV^\top)_{k\nu}} + (VAV^\top)_{\nu k}\frac{P_{\nu k}}{(VAV^\top)_{\nu k}} = \\ &\sum_\nu P_{k\nu} + P_{\nu k}.\end{aligned}$$

On the other hand the sum over i of the right hand side of Equation (2.10) gives

$$\sum_i V_{ki}(\sum_{\lambda\nu} V_{\nu\lambda}A_{i\lambda} + V_{\nu\lambda}A_{\lambda i}) =$$
$$\sum_\nu (VAV^\top)_{k\nu} + (VAV^\top)_{\nu k}.$$

This proves the first part of the theorem. The second part of the theorem follows by summing the left and right hand side of (2.8) over k. ■

A point $(\tilde{A}, \tilde{V})$ is called *normalized* if the matrix $\tilde{V}$ is column stochastic, i.e. $\sum_k \tilde{V}_{ki} = 1, i = 1, 2, \ldots, a$ and $\tilde{A}$ has the same element sum as P, i.e. $\sum_{ij} \tilde{A}_{ij} = \sum_{kl} P_{kl}$. As a consequence of Theorem 2.7, every stationary point (A, V) of the divergence can be written in an equivalent normalized form $(\tilde{A}, \tilde{V})$, such that $VAV^\top = \tilde{V}\tilde{A}\tilde{V}^\top$. The matrices $\tilde{A}$ and $\tilde{V}$ are given as function of A and V by

$$\begin{aligned}\tilde{V} &= V\ (\mathrm{diag}(e^\top V))^{-1},\\ \tilde{A} &= (\mathrm{diag}(e^\top V))\ A\ (\mathrm{diag}(e^\top V)).\end{aligned}$$

The fact that the matrices P and $\tilde{A}$ have the same element sum follows from $e^\top Pe = e^\top \tilde{V}\tilde{A}\tilde{V}^\top e = e^\top \tilde{A}e$.

So, for a stationary point (A, V) of the cost function $D_{KL}(P||VAV^\top)$, there exists a *normalized* version $(\tilde{A}, \tilde{V})$ which gives the same approximation (and hence the same cost function value). Therefore, when minimizing $D_{KL}(P||VAV^\top)$ over nonnegative V and A, it suffices to minimize over nonnegative matrices A and V that are normalized. Minimizing over normalized matrices can be done by choosing normalized initial values and by making sure that the update formulas retain the normalization. Choosing normalized initial values is no problem, and the fact that the proposed update formulas retain the normalization is shown in the proof of Theorem 2.8. From now on, we assume that V or A are normalized, without explicitly indicating it with a tilde.

Theorem 2.8. *Assume that the starting values $V^{(0)} \in \mathbb{R}_+^{p\times a}$ and $A^{(0)} \in \mathbb{R}_+^{a\times a}$ are normalized, i.e. $\sum_{ij} A^{(0)}_{ij} = \sum_{kl} P_{kl}$ and $\sum_k V^{(0)}_{ki} = 1, i = 1, 2, \ldots a$. Then the divergence $D_{KL}(P||VAV^\top)$ is nonincreasing under the update rules*

$$A^{(t+1)}_{ij} = A^{(t)}_{ij} \sum_{\mu\nu} V_{\mu i}V_{\nu j}\frac{P_{\mu\nu}}{(VA^{(t)}V^\top)_{\mu\nu}}, \tag{2.11}$$

$$V^{(t+1)}_{ki} = V^{(t)}_{ki}\frac{\sum_{\lambda\nu} \frac{P_{k\nu}}{(V^{(t)}A(V^{(t)})^\top)_{k\nu}}A_{i\lambda}V^{(t)}_{\nu\lambda} + \frac{P_{\nu k}}{(V^{(t)}A(V^{(t)})^\top)_{\nu k}}A_{\lambda i}V^{(t)}_{\nu\lambda}}{\sum_{\lambda\mu\nu} \frac{P_{\mu\nu}}{(V^{(t)}A(V^{(t)})^\top)_{\mu\nu}}A_{i\lambda}V^{(t)}_{\nu\lambda}V^{(t)}_{\mu i} + \frac{P_{\nu\mu}}{(V^{(t)}A(V^{(t)})^\top)_{\nu\mu}}A_{\lambda i}V^{(t)}_{\nu\lambda}V^{(t)}_{\mu i}}. \tag{2.12}$$

In addition, the updated values of A and V are also normalized.

Proof: First note that the update rule for A retains the normalization, since

$$\sum_{ij} A^{(t)}_{ij} \sum_{\mu\nu} V_{\mu i}V_{\nu j}\frac{P_{\mu\nu}}{(VA^{(t)}V^\top)_{\mu\nu}} = \sum_{\mu\nu}(VA^{(t)}V^\top)_{\mu\nu}\frac{P_{\mu\nu}}{(VA^{(t)}V^\top)_{\mu\nu}} = \sum_{\mu\nu} P_{\mu\nu}.$$

Also the update rule for V retains the normalization, since for $i = 1, 2, \dots a$, it holds that

$$\sum_k V_{ki}^{(t)} \frac{\sum_{\lambda\nu} \frac{P_{k\nu}}{(V^{(t)}A(V^{(t)})^\top)_{k\nu}} A_{i\lambda} V_{\nu\lambda}^{(t)} + \frac{P_{\nu k}}{(V^{(t)}A(V^{(t)})^\top)_{\nu k}} A_{\lambda i} V_{\nu\lambda}^{(t)}}{\sum_{\lambda\mu\nu} \frac{P_{\mu\nu}}{(V^{(t)}A(V^{(t)})^\top)_{\mu\nu}} A_{i\lambda} V_{\nu\lambda}^{(t)} V_{\mu i}^{(t)} + \frac{P_{\nu\mu}}{(V^{(t)}A(V^{(t)})^\top)_{\nu\mu}} A_{\lambda i} V_{\nu\lambda}^{(t)} V_{\mu i}^{(t)}} = 1.$$

Next, we prove that the divergence $D_{KL}(P||VAV^\top)$ is nonincreasing under an update for A. Note therefore that the cost function $F_A(A)$

$$\begin{aligned} F_A(A) &= D_{KL}(P||VAV^\top) \\ &= \sum_{\mu\nu} P_{\mu\nu} \log P_{\mu\nu} - P_{\mu\nu} + (VAV^\top)_{\mu\nu} - P_{\mu\nu} \log(VAV^\top)_{\mu\nu}, \end{aligned}$$

can be approximated by the auxiliary function $G_A(A, A^{(t)})$ given by

$$\begin{aligned} G_A(A, A^{(t)}) &= \sum_{\mu\nu} P_{\mu\nu} \log P_{\mu\nu} - P_{\mu\nu} + (VAV^\top)_{\mu\nu} \\ &\quad - \sum_{\kappa\lambda} P_{\mu\nu} \frac{V_{\mu\kappa} A_{\kappa\lambda}^{(t)} V_{\nu\lambda}}{(VA^{(t)}V^\top)_{\mu\nu}} \left(\log V_{\mu\kappa} A_{\kappa\lambda} V_{\nu\lambda} - \log \frac{V_{\mu\kappa} A_{\kappa\lambda}^{(t)} V_{\nu\lambda}}{(VA^{(t)}V^\top)_{\mu\nu}} \right). \end{aligned}$$

Convexity of the $-\log$ function and Jensen's inequality with $\sum_{\kappa\lambda} \frac{V_{\mu\kappa} A_{\kappa\lambda}^{(t)} V_{\nu\lambda}}{(VA^{(t)}V^\top)_{\mu\nu}} = 1$ gives $G_A(A, A^{(t)}) \geq F_A(A)$. In addition $G_A(A^{(t)}, A^{(t)}) = F_A(A^{(t)})$. To obtain an update formula, we put $A^{(t+1)}$ equal to the minimizer of $G_A(A, A^{(t)})$. From

$$\frac{\partial G_A}{\partial A_{ij}}(A, A^{(t)}) = \sum_{\mu\nu} V_{\mu i} V_{\nu k} - \sum_{\mu\nu} P_{\mu\nu} \frac{V_{\mu i} A_{ij}^{(t)} V_{\nu j}}{(VA^{(t)}V^\top)_{\mu\nu}} \frac{1}{A_{ij}} = 0,$$

we obtain

$$A_{ij}^{(t+1)} = A_{ij}^{(t)} \frac{\sum_{\mu\nu} V_{\mu i} V_{\nu j} \frac{P_{\mu\nu}}{(VA^{(t)}V^\top)_{\mu\nu}}}{\sum_{\mu\nu} V_{\mu i} V_{\nu k}}. \tag{2.13}$$

The denominator is equal to $(\sum_\mu V_{\mu i})(\sum_\nu V_{\nu k}) = 1$, and hence we obtain the proposed update formula (2.11). From

$$\begin{aligned} F_A(A^{(t+1)}) \leq G_A(A^{(t+1)}, A^{(t)}) &= \min_A G_A(A, A^{(t)}) \\ &\leq G_A(A^{(t)}, A^{(t)}) = F_A(A^{(t)}), \end{aligned}$$

it follows that the divergence $D_{KL}(P||VAV^\top)$ is nonincreasing under an update of A.

We now prove that the divergence $D_{KL}(P||VAV^\top)$ is also nonincreasing under an update for V. In order to see this, note that the cost function $F_V(V)$

$$F_V(V) = D_{KL}(P||VAV^\top)$$

$$= \sum_{\mu\nu} P_{\mu\nu}\log P_{\mu\nu} - P_{\mu\nu} + (VAV^\top)_{\mu\nu} - P_{\mu\nu}\log(VAV^\top)_{\mu\nu},$$

can be approximated by the auxiliary function $G_V(V,V^{(t)})$

$$\begin{aligned} G_V(V,V^{(t)}) = &\sum_{\mu\nu} P_{\mu\nu}\log P_{\mu\nu} - P_{\mu\nu} + (V\bar{A}(V^{(t)})^\top)_{\mu\nu} + (V^{(t)}\bar{A}V^\top)_{\mu\nu} - (V^{(t)}\bar{A}(V^{(t)})^\top)_{\mu\nu} \\ &-\sum_{\kappa\lambda} P_{\mu\nu}\frac{V^{(t)}_{\mu\kappa}A_{\kappa\lambda}V^{(t)}_{\nu\lambda}}{(V^{(t)}A(V^{(t)})^\top)_{\mu\nu}}\left(\log V_{\mu\kappa}A_{\kappa\lambda}V_{\nu\lambda} - \log\frac{V^{(t)}_{\mu\kappa}A_{\kappa\lambda}V^{(t)}_{\nu\lambda}}{(V^{(t)}A(V^{(t)})^\top)_{\mu\nu}}\right), \end{aligned}$$

where $\bar{A}$ is chosen later on. At this moment it suffices to require that $\sum_{ij}\bar{A}_{ij} = \sum_{kl}P_{kl}$. From the fact that $\sum_k V_{ki} = \sum_k V^{(t)}_{ki} = 1$ and $\sum_{ij}A_{ij} = \sum_{ij}\bar{A}_{ij} = \sum_{kl}P_{kl}$, we have that

$$\sum_{\mu\nu}(VAV^\top)_{\mu\nu} = \sum_{\mu\nu}(V\bar{A}(V^{(t)})^\top)_{\mu\nu} + (V^{(t)}\bar{A}V^\top)_{\mu\nu} - (V^{(t)}\bar{A}(V^{(t)})^\top)_{\mu\nu}.$$

From this, the convexity of the $-\log$ function and Jensen's inequality with $\sum_{\kappa\lambda}\frac{V^{(t)}_{\mu\kappa}A_{\kappa\lambda}V^{(t)}_{\nu\lambda}}{(V^{(t)}A(V^{(t)})^\top)_{\mu\nu}} = 1$, we obtain that $G_V(V,V^{(t)}) \geq F_V(V)$. In addition, it holds that $G_V(V^{(t)},V^{(t)}) = F_V(V^{(t)})$. To obtain an update formula, we put $V^{(t+1)}$ equal to the minimizer of $G_V(V,V^{(t)})$. From

$$\begin{aligned} \frac{\partial G_V}{\partial V_{ki}}(V,V^{(t)}) &= \sum_{\lambda\nu}\bar{A}_{i\lambda}V^{(t)}_{\nu\lambda} + \bar{A}_{\lambda i}V^{(t)}_{\nu\lambda} - P_{k\nu}\frac{V^{(t)}_{ki}A_{i\lambda}V^{(t)}_{\nu\lambda}}{(V^{(t)}A(V^{(t)})^\top)_{k\nu}}\frac{1}{V_{ki}} \\ &\quad -P_{\nu k}\frac{V^{(t)}_{\nu\lambda}A_{\lambda i}V^{(t)}_{ki}}{(V^{(t)}A(V^{(t)})^\top)_{\nu k}}\frac{1}{V_{ki}} = 0, \end{aligned}$$

we find that

$$V^{(t+1)}_{ki} = V^{(t)}_{ki}\frac{\sum_{\nu\lambda}A_{i\lambda}V^{(t)}_{\nu\lambda}\frac{P_{k\nu}}{(V^{(t)}A(V^{(t)})^\top)_{k\nu}} + A_{\lambda i}V^{(t)}_{\nu\lambda}\frac{P_{\nu k}}{(V^{(t)}A(V^{(t)})^\top)_{\nu k}}}{\sum_{\lambda\nu}\bar{A}_{i\lambda}V^{(t)}_{\nu\lambda} + \bar{A}_{\lambda i}V^{(t)}_{\nu\lambda}}. \tag{2.14}$$

One can easily see that the denominator is equal to $\sum_\lambda \bar{A}_{i\lambda} + \bar{A}_{\lambda i}$. By taking

$$\bar{A}_{ij} = A_{ij}\sum_{\mu\nu}V^{(t)}_{\mu i}V^{(t)}_{\nu j}\frac{P_{\mu\nu}}{(V^{(t)}A(V^{(t)})^\top)_{\mu\nu}}, \tag{2.15}$$

we obtain the proposed update formula for V. From

$$\begin{aligned} F_V(V^{(t+1)}) \leq G_V(V^{(t+1)},V^{(t)}) &= \min_V G_V(V,V^{(t)}) \\ &\leq G_V(V^{(t)},V^{(t)}) = F_V(V^{(t)}). \end{aligned}$$

it follows that the divergence $D_{KL}(P||VAV^\top)$ is nonincreasing under an update of V. ■

We have proven that the divergence $D_{KL}(P||VAV^\top)$ is nonincreasing under the update rules (2.11) and (2.12). We now consider the invariant points of the update formulas and investigate their relation with the stationary points of the divergence $D_{KL}(P||VAV^\top)$.

For fixed $V^{(t)} = V^{(0)}$, the divergence $F(A,V) = D_{KL}(P||VAV^\top)$ is invariant under an update of A, i.e. $A^{(t+1)} = A^{(t)}$, if and only if $A^{(t)}$ is a stationary point of the divergence with fixed $V^{(0)}$, i.e. $A^{(t)}_{ij}\frac{\partial F}{\partial A_{ij}}(A^{(t)},V^{(0)}) = 0$. For fixed $A^{(t)} = A^{(0)}$ on the other hand, the divergence is invariant under an update for V, i.e. $V^{(t+1)} = V^{(t)}$, if and only if

$$\begin{aligned} V^{(t)}_{ki}\Bigg(\sum_{\nu\lambda} \bar{A}^{(0)}_{i\lambda} + \bar{A}^{(0)}_{\lambda i} - A^{(0)}_{i\lambda}V^{(t)}_{\nu\lambda}\frac{P_{k\nu}}{(V^{(t)}A^{(0)}(V^{(t)})^\top)_{k\nu}} & \\ -A^{(0)}_{\lambda i}V^{(t)}_{\nu\lambda}\frac{P_{\nu k}}{(V^{(t)}A^{(0)}(V^{(t)})^\top)_{\nu k}}\Bigg) &= 0. \end{aligned}$$

Notice that this last condition is in general not equivalent to the condition that $V^{(t)}$ is a stationary point of the divergence with fixed $A^{(0)}$. So for the case where we take A fixed and update only V, it is possible that the formulas, if they converge, converge to a point that is not a stationary point of the divergence with fixed $A^{(0)}$.

However, if we use the update formulas for A and V alternatingly, i.e.

$$(A^{(0)},V^{(0)}) \mapsto (A^{(1)},V^{(0)}) \mapsto (A^{(1)},V^{(1)}) \mapsto (A^{(2)},V^{(1)}) \mapsto (A^{(2)},V^{(2)}) \mapsto \dots,$$

we have the following result.

Theorem 2.9. *The divergence is invariant under updates (2.11) and (2.12) if and only if (A,V) is a stationary point of the divergence, i.e.*

$$\left\{\begin{array}{l} A^{(t+1)} = A^{(t)}, \\ V^{(t+1)} = V^{(t)}, \end{array}\right. \Leftrightarrow \left\{\begin{array}{ll} A^{(t)}_{ij}\frac{\partial F}{\partial A_{ij}}(A^{(t)},V^{(t)}) = 0, & i = 1,2,\dots a; j = 1,2,\dots a, \\ V^{(t)}_{ki}\frac{\partial F}{\partial V_{ki}}(A^{(t)},V^{(t)}) = 0, & k = 1,2,\dots p; i = 1,2,\dots a. \end{array}\right.$$

Proof: We first prove the $\Leftarrow$ part. From the fact that $A^{(t)}_{ij}\frac{\partial F}{\partial A_{ij}}(A^{(t)},V^{(t)}) = 0$ for $i = 1,2,\dots a; j = 1,2,\dots a$, it follows that for a certain i,j either $A^{(t)}_{ij} = 0$ or $\frac{\partial F}{\partial A_{ij}}(A^{(t)},V^{(t)}) = 0$. In the first case, the updated value $A^{(t+1)}_{ij}$ is also equal to 0 because the update is multiplicative. In the second case, the update factor for A_{ij} is equal to 1. So, in both cases, we have $A^{(t+1)}_{ij} = A^{(t)}_{ij}$. It also follows that $A^{(t+1)}_{ij}\frac{\partial F}{\partial A_{ij}}(A^{(t+1)},V^{(t)}) = 0$, from which we conclude that $\bar{A}^{(t+1)}_{ij} = A^{(t+1)}_{ij}$. Since $V^{(t)}_{ki}\frac{\partial F}{\partial V_{ki}}(A^{(t)},V^{(t)}) = 0$ for $i = 1,2,\dots a; j = 1,2,\dots a$, we either have $V^{(t)}_{ki} = 0$, or $\frac{\partial F}{\partial V_{ki}}(A^{(t+1)},V^{(t)}) = 0$. In the first case the updated value $V^{(t+1)}_{ki}$ is also equal to 0. In the second case, one can see from $\bar{A}^{(t+1)}_{ij} = A^{(t+1)}_{ij}$ and $\frac{\partial F}{\partial V_{ki}}(A^{(t+1)},V^{(t)}) = 0$, that the update factor for V_{ki} is equal to 1. So in both cases we have that $V^{(t+1)}_{ki} = V^{(t)}_{ki}$. This proves the first part of the theorem.

Next, we prove the $\Rightarrow$ part. From the fact that $A^{(t+1)} = A^{(t)}$, we conclude that either $A_{ij}^{(t)} = 0$ or the update factor for A_{ij} is equal to 1. This implies that $A_{ij}^{(t)} \frac{\partial F}{\partial A_{ij}}(A^{(t)}, V^{(t)}) = 0$. From $A_{ij}^{(t+1)} \frac{\partial F}{\partial A_{ij}}(A^{(t+1)}, V^{(t)}) = 0$, we obtain $\bar{A}_{ij}^{(t+1)} = A_{ij}^{(t+1)}$. From this and $V^{(t+1)} = V^{(t)}$, we conclude that $V_{ki}^{(t)} \frac{\partial F}{\partial V_{ki}}(A^{(t+1)}, V^{(t)}) = 0$. This proves the second part of the theorem. ■

It follows that if the iterates converge, they converge to a stationary point of the cost function in case A and V are updated alternatingly or in case V is fixed and A is updated. However, when A is fixed and only V is updated, it is only guarantueed that the divergence is nonincreasing. It is possible that the formulas converge to a point that is not a stationary point of the divergence with fixed A.

The algorithm below implements the same update formulas as in Theorem 2.8, but is better from computational point of view as the denominator of Equation (2.12) does not have to be computed.

Algorithm 2.1. *Choose arbitrary matrices* $A^{(0)} \in \mathbb{R}_+^{a\times a}$ *and* $V^{(0)} \in \mathbb{R}_+^{p\times a}$ *with* $\sum_{ij} A_{ij}^{(0)} = \sum_{kl} P_{kl}$ *and* $\sum_k V_{ki}^{(0)} = 1, i = 1, 2, \ldots a$. *Now, iterate the following steps for* $t = 0, 1, \ldots$ *until convergence:*

1. $A_{ij}^{(t+1)} = A_{ij}^{(t)} \sum_{\mu\nu} V_{\mu i}^{(t)} V_{\nu j}^{(t)} \frac{P_{\mu\nu}}{(V^{(t)} A^{(t)} (V^{(t)})^\top)_{\mu\nu}}$,

2. $V_{ki}^{(t+1)} = V_{ki}^{(t)} \sum_{\lambda\nu} \frac{P_{k\nu}}{(V^{(t)} A^{(t+1)} (V^{(t)})^\top)_{k\nu}} A_{i\lambda}^{(t+1)} V_{\nu\lambda}^{(t)}$
$$+ \frac{P_{\nu k}}{(V^{(t)} A^{(t+1)} (V^{(t)})^\top)_{\nu k}} A_{\lambda i}^{(t+1)} V_{\nu\lambda}^{(t)},$$

3. *Normalize* $V^{(t+1)}$ *such that* $e^\top V^{(t+1)} = e^\top$.

In case P is symmetric, i.e. $P = P^\top$ and $VAV^\top$ is an approximation of P then one can easily see that $\frac{VAV^\top + (VAV^\top)^\top}{2}$ is a better (or equally good) approximation of P. So if the matrix P is symmetric, we can restrict our search to symmetric approximations. On the other hand, every symmetric approximation $VAV^\top$ can be transformed to a form where A is symmetric by taking $V(\frac{A+A^\top}{2})V^\top$. So in case P is symmetric, we can restrict our search to approximations with $A = A^\top$. This restriction is easy to fulfill in practice since the update formula for A (formula (2.11)) retains the symmetry. So by starting with a symmetric $A^{(0)} = (A^{(0)})^\top$, we end up with a symmetric A.

2.3.2 Application to clustering based on distance matrices

In this section we show that problem where data points are to be *clustered* [59] can be solved using the structured nonnegative matrix factorization. Given p points $y_1, y_2, \ldots y_p$ in $\mathbb{R}^n$, the distance matrix P between the points is given by $P_{kl} = D(y_k, y_l)$ where D is a distance measure on $\mathbb{R}^n$. Note that P is a symmetric matrix and that the diagonal elements of P are equal to 0. Now the clustering problem can be stated as: given the distance matrix P between the points $y_1, y_2, \ldots y_p$ in $\mathbb{R}^n$, find *clusters* of points which are close to each other according to the distance measure D on $\mathbb{R}^n$.

A clustering with a clusters $\{C_1, C_2, \ldots, C_a\}$ is a partition of the set $\{y_1, y_2, \ldots y_p\}$ into disjoint subsets. A clustering is completely described by the matrix $V \in \{0,1\}^{p \times a}$ defined as

$$V_{k,i} = \begin{cases} 1, & \text{if } y_k \in C_i, \\ 0, & \text{if } y_k \notin C_i. \end{cases}$$

Since every point of $\{y_1, y_2, \ldots y_p\}$ belongs to exactly one cluster, we have $Ve = e$.

Define the *mean distance* A_{ij} between the clusters C_i and C_j as the mean of the distances between every possible combination of a point of C_i and a point of C_j.

$$A_{ij} = \frac{\sum_{y_k \in C_i} \sum_{y_l \in C_j} D(y_k, y_l)}{\sum_{y_k \in C_i} \sum_{y_l \in C_j} 1}.$$

It follows that the matrix A with A_{ij} as i, j-th element can be calculated as

$$A = V^{\dagger} P V^{\dagger \top},$$

where $V^{\dagger} = (\mathrm{diag}(e^{\top} V))^{-1} V^{\top}$ is the left inverse of V. Notice that the diagonal elements A_{ii} are equal to the mean distance of the points inside cluster C_i. These elements are hence not necessarily equal to zero.

As a result of the clustering, the distance P_{kl} between two points y_k and y_l is approximated by the mean distance $\tilde{P}_{kl}$ between the clusters to which the points y_k and y_l belong. The complete matrix $\tilde{P}$ can be written as

$$\tilde{P} = VAV^{\top}.$$

From the above, we conclude that the clustering of p points with distance matrix P into a clusters, can be expressed as the following matrix factorization problem

$$\begin{array}{ll} \text{minimize} & C(P, VV^{\dagger}\ P\ (VV^{\dagger})^{\top}) \\ \text{subject to} & V \in \{0,1\}^{p \times a} \\ & Ve = e, \end{array}$$

where $C(X, Y)$ is a divergence/distance measure between the matrices X and Y. As this problem is hard to solve for arbitrary C, we propose the following relaxed version

$$\begin{array}{ll} \text{minimize} & D_{KL}(P \| VAV^{\top}) \\ \text{subject to} & V \in \mathbb{R}_+^{p \times a}, \quad A = A^{\top} \in \mathbb{R}_+^{a \times a}. \end{array}$$

This problem can be solved with the methods proposed in Section 2.3.1. A point y_k is assigned to cluster i where $i = \mathrm{argmax}_t V_{kt}$. By using this relaxed version of the problem, we have two additional advantages. First of all the quality of the clustering can be measured by the ratio between the diagonal and off-diagonal elements of A. The smaller the diagonal elements of A compared to the off-diagonal elements of A, the better the clustering. In addition, one has a measure for how strong a certain point belongs to its cluster. We say that y_k belongs to cluster i if V_{ki} is the biggest element of row $V_{k,:}$. If all other elements of the row $V_{k,:}$ are much smaller than V_{ki}, one can say that the point y_k strongly belongs to cluster i. If there are elements in the row $V_{k,:}$ that are of the same order of magnitude as V_{ki}, we conclude that the point weakly belongs to cluster i.

We now apply this algorithm to a data set of iris flowers (this data set is available in the SOM-toolbox for Matlab as the file `iris.mat`). The data set contains 150 data points $y_1, \dots y_{150}$. Each point contains 4 measurements of an iris flower. The 4 measurements are the petal width, petal length, sepal width and sepal length of the flower. In the data set three different types of flowers are present, the first 50 samples are Setosa flowers, the next 50 are Versicolor flowers and the last 50 are Virginica flowers. As a first step we make a distance matrix P of size 150×150 with $P_{kl} = ||y_k - y_l||_2$. Next we decompose the matrix P into a product $VAV^\top$. As we do not know the number of clusters in advance, we make decompositions with inner dimensions 2 to 6. In Figure 2.1 we show the distance between the original matrix P and its approximation $VAV^\top$ as a function of the structured positive rank of the approximation. One sees that the divergence does not decrease much by taking the structured positive rank higher than 3. For that reason, we conclude to work with 3 clusters.

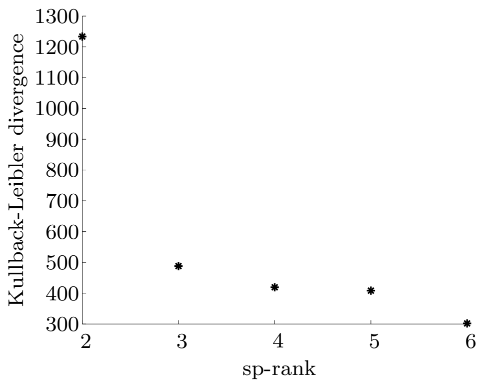


Figure 2.1: *Kullback-Leibler divergence between the true distance matrix P and its optimal (w.r.t. the Kullback-Leibler divergence) approximation of structured positive rank $2, 3, \dots, 6$ computed with the iterative algorithm of Section 2.3.1.*

The A-matrix of the decomposition is given by

$$A = \begin{bmatrix} 0,00000000001 & 4975,78659538 & 9668,80064875 \\ 4975,78659538 & 0,00000032641 & 13768,5062877 \\ 9668,80064875 & 13768,5062877 & 27,0549877873 \end{bmatrix}.$$

Notice that the diagonal elements of A are small compared to the off-diagonal elements of A, which is an indication that we have a good clustering. In addition, the diagonal elements give a measure of the density of the clusters. We conclude that the first cluster is the densest while the third cluster is the least dense. The off-diagonal elements of A give an idea of the mean distance between the clusters.

As explained before, the largest element of the k-th row of the matrix V allows to conclude to which cluster point k belongs. Using this approach 136 of the 150 iris flowers are clustered correctly (i.e. Versicolor flowers in cluster 1, Virginica flowers in cluster 2 and Setosa flowers in cluster 3). Moreover, the elements of $V_{k,:}$ also give a measure of the strength with which the point belong to its cluster. In Table 2.1, we show the k-th row of the matrix V for $k = 14, 53, 58, 130$. For instance for point y_{14} we conclude from our algorithm that it strongly belongs to cluster 3 (as $0.196 \gg 0.017$ and $0.196 \gg 0.012$). This makes sense as the true cluster of that point is indeed cluster 3. On the other hand, point y_{53} (which is one of the miss-clustered points) was connected to cluster 2 but the connection is not strong, its affinity with cluster 1 is almost as high as its affinity with cluster 2. This again makes sense as the true cluster of this point was cluster 1. In Figure 2.2, we visualize the result of the clustering algorithm. Points that belong to cluster 1 are plotted with o, point belonging to cluster 2 are plotted with $*$, and points belonging to cluster 3 are plotted with $+$.

Table 2.1: *Clustering result for the points y_{14}, y_{53}, y_{58} and y_{130}.*

Point y_k	True cluster	$V(k,1)$	$V(k,2)$	$V(k,3)$	Estimated cluster
14	3	0.017	0.012	0.196	3
53	1	0.080	0.115	0.001	2
58	1	0.123	0.000	0.071	1
130	2	0.002	0.201	0.015	2

2.4 Symmetric nonnegative matrix factorization

In the *exact symmetric nonnegative matrix factorization problem*, we are given a square, symmetric, nonnegative definite matrix $P \in \mathbb{R}_+^{p\times p}$, and are looking for a decomposition $P = VV^\top$ with $V \in \mathbb{R}_+^{p\times a}$. The completely positive rank (cp–rank) [9, 14, 15] of the matrix P is the minimal inner dimension for which a decomposition $P = VV^\top$ exists. In contrast to the nonnegative matrix factorization and the structured nonnegative matrix factorization, a symmetric nonnegative factorization with a finite inner dimension does not always exist. In case the symmetric decomposition does not exist, we say that the completely positive rank is infinite. Again, from the definition of the different ranks, it follows that $0 \leq \operatorname{rank} P \leq \text{p–rank}\, P \leq \text{sp–rank}\, P \leq \text{cp–rank}\, P$. It is shown in [14] that there exists a finite algorithm to compute the completely positive

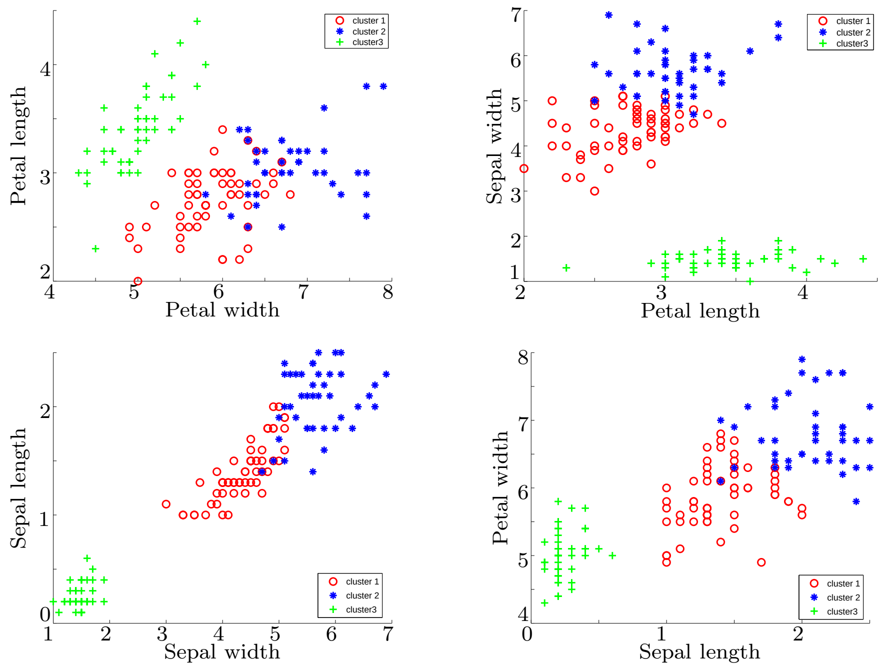


Figure 2.2: *Visualisation of the result of our clustering algorithm. Points that belong to cluster 1 are plotted with* o*, point belonging to cluster 2 are plotted with* $*$*, and points belonging to cluster 3 are plotted with* $+$*.*

rank. However, the complexity bounds of this algorithm are non-polynomial. Therefore *(approximate) symmetric nonnegative matrix factorization problem* has been introduced.

Problem 2.3. *Given a symmetric matrix* $P \in \mathbb{R}_+^{p \times p}$ *and given* a*, minimize* $D_{KL}(P||VV^\top)$ *with respect to* V *(of size* $p \times a$*), subject to the constraint* $V \geq 0$.

Analogous to the decomposition $VAV^\top$, one can prove that the row (or column) sum of P is equal to the row (or column) sum of $VV^\top$, where V is a stationary point of the divergence $D_{KL}(P||VV^\top)$. As a consequence the element sum of P is equal to the element sum of $VV^\top$ with V a stationary point of the divergence.

Theorem 2.10. *Given a nonnegative matrix* $P \in \mathbb{R}^{p \times p}$*. Then every stationary point* V *of the cost function* $D_{KL}(P||VV^\top)$ *preserves the element sum of* P*, i.e.*

$$\sum_{kl} P_{kl} = \sum_{kl} (VV^\top)_{kl}.$$

As a consequence of this theorem, we see that for every stationary point V of the divergence $D_{KL}(P||VV^\top)$, there exists a matrix $\tilde{V}$ and a diagonal matrix D such that $VV^\top = \tilde{V}D\tilde{V}^\top$, where $\tilde{V}$ is column stochastic, i.e. $\sum_k \tilde{V}_{ki} = 1, i = 1, 2, \dots, a$ and the element sum of P equals the element sum of D, i.e. $\sum_{kl} P_{kl} = \sum_i D_{ii}$.

Our approach for Problem 2.3 is to look for a decomposition $P \simeq \tilde{V}D\tilde{V}^\top$, with $\tilde{V}$ column stochastic and D diagonal with sum of its elements equal to the element sum of P such that the divergence $F(D, \tilde{V}) = D_{KL}(P||\tilde{V}D\tilde{V}^\top)$ is minimized. This leads to updates which make the divergence decrease, and are invariant if and only if we have reached a stationary point of the divergence $D_{KL}(P||\tilde{V}D\tilde{V}^\top)$. Once an approximation $P \simeq \tilde{V}D\tilde{V}^\top$ is found, we obtain a decomposition of the form $VV^\top$, by calculating V as

$$V = \tilde{V}\sqrt{D}.$$

The next theorem proposes update formulas for the decomposition $P = \tilde{V}D\tilde{V}^\top$.

Theorem 2.11. *Under the condition that the starting values $\tilde{V}^{(0)}$ and $D^{(0)}$ are normalized, i.e. $\sum_i D_{ii}^{(0)} = \sum_{kl} P_{kl}$ and $\sum_k \tilde{V}_{ki}^{(0)} = 1, i = 1, 2, \dots a$, the divergence $D_{KL}(P||\tilde{V}D\tilde{V}^\top)$ is nonincreasing under the update rules*

$$D_{ii}^{(t+1)} = D_{ii}^{(t)} \sum_{\mu\nu} \tilde{V}_{\mu i}\tilde{V}_{\nu i} \frac{P_{\mu\nu}}{(\tilde{V}D^{(t)}\tilde{V}^\top)_{\mu\nu}}, \tag{2.16}$$

$$\tilde{V}_{ki}^{(t+1)} = \tilde{V}_{ki}^{(t)} \frac{\sum_\nu \frac{P_{k\nu}}{(\tilde{V}^{(t)}D(\tilde{V}^{(t)})^\top)_{k\nu}} D_{ii}\tilde{V}_{\nu i}^{(t)} + \frac{P_{\nu k}}{(\tilde{V}^{(t)}D(\tilde{V}^{(t)})^\top)_{\nu k}} D_{ii}\tilde{V}_{\nu i}^{(t)}}{\sum_{\mu\nu} \frac{P_{\mu\nu}}{(\tilde{V}^{(t)}D(\tilde{V}^{(t)})^\top)_{\mu\nu}} D_{ii}\tilde{V}_{\nu i}^{(t)}\tilde{V}_{\mu i}^{(t)} + \frac{P_{\nu\mu}}{(\tilde{V}^{(t)}D(\tilde{V}^{(t)})^\top)_{\nu\mu}} D_{ii}\tilde{V}_{\nu i}^{(t)}\tilde{V}_{\mu i}^{(t)}}. \tag{2.17}$$

In addition, the updated values of D and $\tilde{V}$ are also normalized.

Proof: The proof is analogous to the proof of Theorem 2.8. ■

If we use the update formulas for D and $\tilde{V}$ alternatingly, i.e.

$$(D^{(0)}, \tilde{V}^{(0)}) \mapsto (D^{(1)}, \tilde{V}^{(0)}) \mapsto (D^{(1)}, \tilde{V}^{(1)}) \mapsto (D^{(2)}, \tilde{V}^{(1)}) \mapsto (D^{(2)}, \tilde{V}^{(2)}) \mapsto \dots,$$

we can prove the following theorem.

Theorem 2.12. *The divergence is invariant under updates (2.16) and (2.17) if and only if $(D, \tilde{V})$ is a stationary point of the divergence, i.e.*

$$\left\{ \begin{array}{l} D^{(t+1)} = D^{(t)}, \\ \tilde{V}^{(t+1)} = \tilde{V}^{(t)}, \end{array} \right. \Leftrightarrow \left\{ \begin{array}{ll} D_{ii}^{(t)} \frac{\partial F}{\partial D_{ii}}(D^{(t)}, \tilde{V}^{(t)}) = 0, & i = 1, 2, \dots a, \\ \tilde{V}_{ki}^{(t)} \frac{\partial F}{\partial \tilde{V}_{ki}}(D^{(t)}, \tilde{V}^{(t)}) = 0, & k = 1, 2, \dots, p; i = 1, 2, \dots a. \end{array} \right.$$

Proof: The proof is analogous to the proof of Theorem 2.9. ■

2.5 Nonnegative matrix factorization without nonnegativity constraints on the factors

In this section we introduce and solve the nonnegative matrix factorization problem without nonnegativity constraints on the factors. This matrix factorization is studied because of its importance for the approximate quasi realization problem for HMMs (Section 4.3). However we show that the factorization has applications apart from hidden Markov models.

In Section 2.5.1 we give an intuitive motivation to the problem. In Section 2.5.2 we formally introduce the factorization problem and provide an algorithm to solve it. In Section 2.5.3 we show that the approach allows to impose upper bounds on the elements of the low rank approximation instead of only lower bounds. In Section 2.5.4 finally, we apply the method to an image compression example.

2.5.1 Intuitive motivation

In the paper of Lee and Sueng [72] the nonnegative matrix factorization is used to model a database of facial images. Each face in the database is represented by a nonnegative column vector, leading to a nonnegative matrix with the number of columns equal to the number of faces in the database. Subsequently, the matrix is factored as a product VH with the matrices V and H nonnegative. The columns of V are called *eigenfaces* and the columns of H are nonnegative weights that reconstruct the original faces as linear combinations of the eigenfaces. By carrying out an approximation of M as VH with V and H nonnegative, an efficient compression of the original database can be obtained. A first argument for using the nonnegative matrix factorization instead of the SVD or other matrix decompositions is that the reconstructed images VH contain only nonnegative elements and are therefore interpretable as facial images. A second argument is that the eigenfaces are typically sparse (i.e. many elements are zero) because of the nonnegativity of the weights in H, which makes that there is a *part-based* interpretation of decomposition, i.e. each face is decomposed into a sum of a small number of *parts* (nose, eyes, ears, ...). In our opinion, the first argument is always relevant, while the second is not important in certain applications (for instance in the application where the decomposition is only used for data compression). We therefore consider the decomposition of a nonnegative matrix M into a nonnegative product VH, but where the factors V and H themselves are allowed to contain negative values. In this section we give an intuitive example motivating our approach.

Suppose one wants to model moustaches and beards of a database of human faces with 4 parameters. The first parameter represents the top of the moustache (closest to the nose) and the second the bottom of the moustache. The third parameters is used for the top of the beard (closest to the mouth) and the fourth for the bottom of the beard (see Figure 2.3(a)). The values for the parameters range from 0 (no moustache/beard) over 0.5 (light-coloured moustache/beard) to 1 (dark moustache/beard). In Figure 2.3 (b)-(e), we show faces with four different possible combinations of dark beards and moustaches.

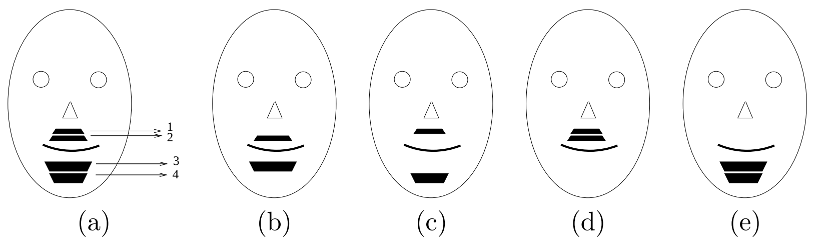


Figure 2.3: *(a) Interpretation of the 4 parameters used to model human moustaches and beards, (b)-(e) Four different examples of combinations of beard and moustaches.*

The matrix M, which contains in its i-th column the 4 parameters of face i is given by

$$M = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

This matrix M has rank 3, but positive rank 4 (see [33]). This means that there exists an exact decomposition of inner dimension 3 if negative factors V and H are allowed, but that the smallest inner dimension for a decomposition with nonnegative factors is equal to 4, i.e.

$$\begin{aligned} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} &= \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ -1 & 1 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

Of course, one can also build an approximate decomposition of inner dimension 3 with nonnegative factors,

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \simeq \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{2} \\ 0 & \frac{2}{3} & 0 \\ \frac{2}{3} & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 2 \\ 1 & 0 & 2 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & 1 & \frac{2}{3} & 0 \\ \frac{2}{3} & 0 & \frac{4}{3} & 0 \\ \frac{2}{3} & 0 & 0 & \frac{4}{3} \\ \frac{1}{3} & 1 & 0 & \frac{2}{3} \end{bmatrix}.$$

With this academic example, we show that by allowing negative elements in the factors of the decomposition, we can find an exact decomposition of lower rank than the decomposition that is found when only nonnegative factors are allowed. Here it is clear that the decomposition with negative factors gives rise to a nonnegative product VH as the decomposition is exact.

In Section 2.5.2 we propose a decomposition method for which VH is nonnegative (but V and H not) even if the decomposition is not exact. It is expected that this decomposition finds an approximation of the same quality as the NMF with a smaller inner dimension (so a larger data reduction) or finds a better approximation if the same inner dimension is used as for the NMF.

In Section 2.5.4 we illustrate that the effects illustrated in this intuitive example extend to larger matrix decomposition problems.

2.5.2 Nonnegative matrix factorization without nonnegativity constraints on the factors

The approximate nonnegative matrix factorization without nonnegativity constraints on the factors can be stated as

Problem 2.4. *Given $M \in \mathbb{R}_+^{m_1 \times m_2}$ and $a \in \mathbb{N}$, minimize $D_{\mathrm{KL}}(M||VH)$ with respect to V $(\in \mathbb{R}^{m_1 \times a})$ and H $(\in \mathbb{R}^{a \times m_2})$, subject to the constraint $VH \geq 0$.*

Notice that the solution of Problem 2.4 is not unique. If (V, H) is a solution the problem, then $(VT, T^{-1}H)$, with T a nonsingular matrix in $\mathbb{R}^{a \times a}$ is also a solution. Next, it turns out that the constraint $VH \geq 0$ is automatically fullfilled by using the Kullback-Leibler divergence as distance criterion. Indeed, the Kullback-Leibler divergence $D_{\mathrm{KL}}(M||VH)$ goes to infinity if an element of VH goes to 0, while the corresponding element of M is not equal to 0, and the divergence is undefined if an element of VH is negative.

The partial derivatives of $F(V, H) = D_{\mathrm{KL}}(M||VH)$ with respect to the elements V_{ki} and H_{il} are

$$\frac{\partial F}{\partial V_{ki}}(V,H) = -\sum_{\nu} H_{i\nu} \frac{M_{k\nu}}{(VH)_{k\nu}} + \sum_{\nu} H_{i\nu}, \tag{2.18}$$

$$\frac{\partial F}{\partial H_{il}}(V,H) = -\sum_{\mu} V_{\mu i} \frac{M_{\mu l}}{(VH)_{\mu l}} + \sum_{\mu} V_{\mu i}, \tag{2.19}$$

Hence the conditions for stationarity of (V, H) with respect to $D_{\mathrm{KL}}(M||VH)$ are

$$\frac{\partial F}{\partial V_{ki}}(V,H) = 0, \qquad \frac{\partial F}{\partial H_{il}}(V,H) = 0, \tag{2.20}$$

for $i = 1, 2, \ldots, a$, $k = 1, 2, \ldots, m_1$ and $l = 1, 2, \ldots, m_2$.

In the case $a = 1$, the global optimal solution to Problem 2.4 can be obtained analytically. Moreover, the global optimal solution (V_1, H_1) is equal to the global optimal solution of Problem 2.1 with $a = 1$ and is given by

$$(V_1)_{k1} = \frac{Me}{\sqrt{e^\top Me}}, \qquad (H_1)_{1l} = \frac{e^\top M}{\sqrt{e^\top Me}}. \tag{2.21}$$

For the case $a > 1$, we propose below an iterative algorithm that converges to a stationary point of F. First, we establish some interesting properties of the stationary points of Problem 2.4. The first property is analogous to the property of the classical nonnegative matrix factorization proven in [57]. It says that the row and column sum of the matrix M is equal to the row and column sum of its optimal approximation.

Proposition 2.1. *Let $M \in \mathbb{R}^{m_1 \times m_2}$. Every stationary point (V, H) of the cost function $D_{\mathrm{KL}}(M||VH)$ preserves the row and column sum of M, i.e.*

$$\begin{aligned} Me &= (VH)e \qquad (2.22)\\ e^\top M &= e^\top (VH) \qquad (2.23) \end{aligned}$$

As a consequence the element sum of M is preserved, i.e. $e^\top Me = e^\top (VH)e$.

Proof: Equation (2.20) gives, for $k = 1, 2, \ldots, m_1$ and $i = 1, 2, \ldots, a$,

$$\sum_\nu H_{i\nu} \frac{M_{k\nu}}{(VH)_{k\nu}} = \sum_\nu H_{i\nu}.$$

Multiplying with V_{ki} and summing over i proves Equation (2.22). The proof of Equation (2.23) is analogous.

■

We now prove a second property of stationary points.

Proposition 2.2. *Let $M \in \mathbb{R}^{m_1 \times m_2}$. Given a stationary point (V, H) of the cost function $D_{\mathrm{KL}}(M||VH)$, there exists a nonsingular matrix $T \in \mathbb{R}^{a \times a}$ such that $\tilde{V} = VT$ and $\tilde{H} = T^{-1}H$ satisfy*

$$\begin{aligned} \tilde{V}_{:,1} &= \frac{Me}{\sqrt{e^\top Me}}, \qquad (2.24)\\ \tilde{H}_{1,:} &= \frac{e^\top M}{\sqrt{e^\top Me}}. \qquad (2.25) \end{aligned}$$

Proof: Define

$$\begin{aligned} V' &= VR,\\ H' &= R^{-1}H, \end{aligned}$$

with $R = \left[\begin{array}{cc} \frac{He}{\sqrt{e^\top Me}} & R' \end{array}\right]$ where R' is such that R is nonsingular. Since $He \neq 0$ such a matrix R' always exists. Then

$$V'_{:,1} = V\frac{He}{\sqrt{e^\top Me}} = \frac{Me}{\sqrt{e^\top Me}}$$

Calculate now

$$\begin{aligned} \tilde{V} &= V'S^{-1} \\ \tilde{H} &= SH' \end{aligned}$$

with $S = \begin{bmatrix} \frac{e^\top V'}{\sqrt{e^\top Me}} \\ \mathbf{0} \quad S' \end{bmatrix}$, where S' is such that S is nonsingular. Since $\frac{e^\top V'}{\sqrt{e^\top Me}}(1) = 1$, such a matrix S' always exists. Hence

$$\tilde{H}_{1,:} = \frac{e^\top V'}{\sqrt{e^\top Me}} H' = \frac{e^\top M}{\sqrt{e^\top Me}}.$$

From $S_{1,1} = \frac{\sqrt{e^\top Me}}{\sqrt{e^\top Me}} = 1$, we obtain that the first column of S^{-1} is equal to $[\, 1\ 0\ 0\ \ldots\ 0]^\top$, and hence

$$\tilde{V}_{:,1} = \frac{Me}{\sqrt{e^\top Me}}.$$

■

A third property of the matrix factorization is related to the nesting property of the singular value decomposition. The nesting property of the SVD says that the optimal rank b (with $b < a$) approximation of the optimal rank a approximation of a given matrix M is equal to the optimal rank b approximation of the original matrix M. In addition the optimal rank a SVD-approximation can be written in a form $M_a = \tilde{V}_a\tilde{H}_a$ such that the optimal rank b approximation is equal to $M_b = V_bH_b$, where $V_b = (\tilde{V}_a)_{:,1:b}$ and $H_b = (\tilde{H}_a)_{1:b,:}$. For the proposed matrix factorization, we have the same property for $b = 1$. Indeed, from Proposition 2.1 and Equation (2.21), we know that the optimal rank 1 approximation $M_1 = V_1H_1$ of an optimal rank a approximation V_aH_a of a matrix M is equal to the optimal rank 1 approximation of the original matrix M. In addition, Proposition 2.2 says that the optimal approximation of rank a can be written in a form $M_a = \tilde{V}_a\tilde{H}_a$ such that $(\tilde{V}_a)_{:,1} = V_1$ and $(\tilde{H}_a)_{1,:} = H_1$.

We now provide an algorithm to solve the nonnegative matrix factorization problem without nonnegativity constraints on the factors. Analogously to the nonnegative matrix factorization problem, the matrix factorization problem (Problem 2.4) is convex in V and H separately, but is not convex in V and H jointly. Therefore, we use an approach where V and H are updated alternatingly. In the proposed algorithm, these updates are performed using one step of the damped Newton method. In a step of the damped Newton method, one first builds a quadratic approximation of the cost function in the current value for the optimization variable. Next, one makes a step in the direction of the minimum of this quadratic function. The step size is determined by a backtracking approach. Formally, a Newton step to optimize the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is given by

$$x^{(t+1)} = x^{(t)} - \mu \left[\frac{\partial^2 f}{\partial x_i \partial x_j}(x^{(t)})\right]^{-1} \left[\frac{\partial f}{\partial x_i}(x^{(t)})\right],$$

where $\left[\frac{\partial f}{\partial x_i}\right]$ is the gradient, $\left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]$ the Hessian of f, and μ is the stepsize.

We first discuss the updating of H. The optimization problem with cost $F(V,H) = D_{\mathrm{KL}}(M||VH)$ splits into m_2 independent problems related to each column of H. We therefore consider the partial cost for a single column of M and H which we denote by $M_{:,l}$ and $H_{:,l}$. The gradient $\in \mathbb{R}^{1\times a}$ of the cost function $F(V,H_{:,l}) = D_{\mathrm{KL}}(M_{:,l}||VH_{:,l})$ is given by (2.19) and the Hessian $\in \mathbb{R}^{a\times a}$ is given by

$$\frac{\partial^2 F}{\partial H_{il}\partial H_{jl}}(V,H) \quad = \quad \sum_{\mu} V_{\mu i} V_{\mu j} \frac{M_{\mu l}}{((VH)_{\mu l})^2}.$$

The update rule for $H_{:,l}$ becomes

$$H_{:,l}^{(t+1)} \quad = \quad H_{:,l}^{(t)} - \mu \left[\frac{\partial^2 F}{\partial H_{il}\partial H_{jl}}(V,H^{(t)})\right]^{-1} \left[\frac{\partial F}{\partial H_{il}}(V,H^{(t)})\right].$$

The updating rule for V is completely analogous. The Hessian of the cost function $F(V_{k,:},H) = D_{\mathrm{KL}}(M_{k,:}||V_{k,:}H)$ is given by

$$\frac{\partial^2 F}{\partial V_{ki}\partial V_{kj}}(V,H) \quad = \quad \sum_{\nu} H_{i\nu} H_{j\nu} \frac{M_{k\nu}}{((VH)_{k\nu})^2}.$$

A formal statement of the complete algorithm is given in Algorithm 2.2. The parameter α needs to be smaller than 1 and can be chosen by the user.

Algorithm 2.2. *Take* $\alpha \in]0,1[$. *Choose arbitrary matrices* $V^{(0)} \in \mathbb{R}_+^{m_1\times a}$ *and* $H^{(0)} \in \mathbb{R}_+^{a\times m_2}$. *Now, iterate the following steps for* $t = 0,1,\dots$ *until convergence:*

1. *Set* $\mu^{(1)} = 1$. *Perform the following steps for* $p = 1,\dots$ *until* $V^{(t+1)}H^{(t)}$ *is nonnegative:*

   (a) $H_{:,l}^{(t+1)} = H_{:,l}^{(t)} - \mu^{(p)} \left[\frac{\partial^2 F}{\partial H_{il}\partial H_{jl}}(V^{(t)},H^{(t)})\right]^{-1} \left[\frac{\partial F}{\partial H_{il}}(V^{(t)},H^{(t)})\right]$,
   $(l = 1,2,\dots m_2)$,

   (b) $\mu^{(p+1)} = \alpha\mu^{(p)}$.

2. *Set* $\nu^{(1)} = 1$. *Perform the following steps for* $q = 1,\dots$ *until* $V^{(t+1)}H^{(t+1)}$ *is nonnegative:*

   (a) $V_{k,:}^{(t+1)} = V_{k,:}^{(t)} - \nu^{(q)} \left[\frac{\partial^2 F}{\partial V_{ki}\partial V_{kj}}(V^{(t)},H^{(t+1)})\right]^{-1} \left[\frac{\partial F}{\partial V_{ki}}(V^{(t+1)},H^{(t)})\right]$,
   $(k = 1,2,\dots m_1)$,

(b) $\nu^{(q+1)} = \alpha\nu^{(q)}$.

In the algorithm the inverse of the Hessian has to be computed. This computation can give rise to two different kind of problems. We describe both problems and shortly discuss a possible solution. The first possible problem is that the Hessian can become singular such that its inverse can not be computed. A solution therefore is to add λI to the Hessian before inverting it, where λ is a to be chosen constant and I is the unit matrix of appropriate size. This approach is known as the Levenberg-Marquardt method. A second problem is that the computation and storage of the Hessian becomes very expensive when decomposing a large-scale matrix. A solution therefore is to use other optimization methods that do not need the Hessian for computing an update, such as gradient or conjugate gradient methods. We do not go into detail about these algorithms.

There exist applications where the nonnegativity of V as well as the nonnegativity of the product VH are important, but where the nonnegativity of H is not important. In that case, one can easily combine the update formula for V of the nonnegative matrix factorization with the update formulas for H of the matrix factorization of this section. A similar construction holds, of course, if H and VH are required to be nonnegative, but not V.

2.5.3 Imposing upper bounds

The nonnegative matrix factorization is used in many applications where the variables are constrained to be nonnegative from physical point of view. However, in many applications, the variables are not only bounded from below, but are also bounded from above. For instance, the greyscale value of an image is usually a number between 0 and 1. Therefore in this application, it is odd to impose the lower bound 0, while not imposing the upper bound 1. The algorithm given in Section 2.5.2 can easily deal with upper bounds. Introduce the modified Kullback-Leibler divergence between the matrices $X \in \mathbb{R}^{x_1\times x_2}$ and $Y \in \mathbb{R}^{x_1\times x_2}$

$$\begin{aligned} D_{\mathrm{MKL}}(X||Y) &:= D_{\mathrm{KL}}(X||Y) + D_{\mathrm{KL}}(\mathbf{1}_{x_1,x_2} - X||\mathbf{1}_{x_1,x_2} - Y) \\ &= \sum_{ij} X_{ij}\log\frac{X_{ij}}{Y_{ij}} + \sum_{ij}(1-X_{ij})\log\frac{1-X_{ij}}{1-Y_{ij}}. \end{aligned}$$

Now consider the following problem

Problem 2.5. *Given $M \in \mathbb{R}_+^{m_1\times m_2}$ and $a \in \mathbb{N}$, minimize $G(V,H) = D_{\mathrm{MKL}}(M||VH)$ with respect to V ($\in \mathbb{R}^{m_1\times a}$) and H ($\in \mathbb{R}^{a\times m_2}$), subject to the constraints $0 \le (VH)_{kl} \le 1$.*

The constraints $0 \le (VH)_{kl} \le 1$ are automatically fullfilled by using the modified version of the Kullback-Leibler divergence. Indeed, the modified Kullback-Leibler divergence is undefined if VH is negative or larger than 1, and in addition, the modified divergence $D_{\mathrm{MKL}}(M||VH)$ goes to infinity if an element of VH goes to 0 or to 1, while the corresponding element of M is not equal to 0 or 1.

The approximate matrix factorization algorithm (Algorithm 2.2) can be easily adapted to deal with Problem 2.5 by using the following expressions for the gradient and Hessian.

$$\begin{aligned}
\frac{\partial G}{\partial V_{ki}}(V,H) &= -\sum_{\nu} H_{i\nu}\frac{M_{k\nu}}{(VH)_{k\nu}} + H_{i\nu}\frac{1-M_{k\nu}}{1-(VH)_{k\nu}},\\
\frac{\partial G}{\partial H_{il}}(V,H) &= -\sum_{\mu} V_{\mu i}\frac{M_{\mu l}}{(VH)_{\mu l}} + V_{\mu i}\frac{1-M_{\mu l}}{1-(VH)_{\mu l}},\\
\frac{\partial^2 G}{\partial V_{ki}\partial V_{kj}}(V,H) &= \sum_{\nu} H_{i\nu}H_{j\nu}\frac{M_{k\nu}}{((VH)_{k\nu})^2} + H_{i\nu}H_{j\nu}\frac{1-M_{k\nu}}{(1-(VH)_{k\nu})^2},\\
\frac{\partial^2 G}{\partial H_{il}\partial H_{jl}}(V,H) &= \sum_{\mu} V_{\mu i}V_{\mu j}\frac{M_{\mu l}}{((VH)_{\mu l})^2} + V_{\mu i}V_{\mu j}\frac{1-M_{\mu l}}{(1-(VH)_{\mu l})^2}.
\end{aligned}$$

2.5.4 Application to image compression

In this application, we use data from the CBCL-database of human faces [1]. This database contains 2429 facial greyscale images of size 19×19. We use the first 100 images, vectorize and stack them in the columns of the matrix M as is done in [72]. We then approximate the matrix M with a product VH with inner dimension $a = 20$ with four different procedures: nonnegative matrix factorization (NMF), the singular value decomposition (SVD), the nonnegative matrix factorization without nonnegativity constraints on the factors (MF), and the nonnegative matrix factorization without nonnegativity constraints on the factors with upper bound 1 on the elements of VH (UBMF). The NMF, MF, UBMF algorithms converged using less than 50 iteration steps, but the results shown are after 100 iterations.

Table 2.2 shows that MF and UBMF give the best results, illustrated by the Kullback-Leibler and the modified Kullback-Leibler divergence between M and the approximation VH computed with the four different methods. For all methods, pixel values of the reconstruction that are ≤ 0 or ≥ 1, are truncated to the values 0.0001 and 0.9999 respectively. We do not truncate these values to 0 and 1, as this operation makes the distances indicated with (*) equal to infinity which makes comparison between the methods impossible.

In Figure 2.4, we plot face 34 and 70 and their reconstructions using the 4 different methods. Pixel values smaller than 0 are indicated with o and pixel values higher than 1 are indicated with $\times$. From Table 2.2 and from Figure 2.4, we conclude that the matrix factorization (MF) and the upper bounded matrix factorization (UBMF) give the best results.

Table 2.2: *Kullback-Leibler divergence and modified Kullback-Leibler divergence between M and the decomposition VH for the 4 different decomposition methods.*

	MF	UBMF	NMF	SVD
KL-divergence	339	383	564	460 (*)
MKL-divergence	1104 (*)	806	1470 (*)	989 (*)

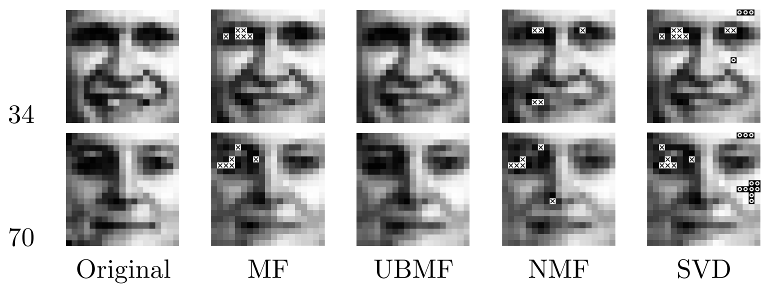


Figure 2.4: *Plot of the i-th face of the database, with $i = 34, 70$, and their reconstructions using the nonnegative matrix factorization without nonnegativity constraints on the factors (MF), the nonnegative matrix factorization without nonnegativity constraints on the factors with upper bounds on the elements of VH (UBMF), the nonnegative matrix factorization (NMF) and the singular value decomposition (SVD). All approximate factorizations have inner dimension $a = 20$. Pixel values smaller than 0 are indicated with o and pixel values higher than 1 are indicated with $\times$.*

2.6 Conclusions

This chapter dealt with low rank approximations of matrices. We reviewed the singular value decomposition and proved a small property of the SVD-truncation of matrices with symmetries. The rank k SVD-truncation of a matrix M that obeys $M = PMQ$ with P and Q unitary, has the same type of symmetry, i.e. $M_k = PM_kQ$.

The approximate nonnegative matrix factorization was reviewed. Subsequently, the approximate structured nonnegative matrix factorization problem was introduced. We proposed iterative update formulas for this problem and proved their convergence properties. The factorization has been applied to a clustering problem where data points are to be clustered based on their distance matrix. The symmetric nonnegative matrix factorization problem is considered as a special case of the structured nonnegative matrix factorization problem.

Finally, the nonnegative matrix factorization without nonnegativity constraints on the factors was introduced. The importance of this problem was explained and update formulas for this problem were derived. The factorization has been applied to the problem of compressing a database of facial images.

Chapter 3

Hidden Markov models - Linear stochastic models

Hidden Markov models have been introduced in the literature in three different-looking forms. The first form is the *HMM of the deterministic function of a Markov chain type* and was introduced in [20, 52]. The second form is the *HMM of the Moore type* and was introduced in the literature in [10]. The third form is the *HMM of the Mealy type* and was introduced in [84]. It has been shown in [112,113] that although the three types of models look quite different, they are all equivalent from the viewpoint of their "expressive power", meaning that a process is representable by a HMM of a certain form if and only if it is representable by a HMM of another form.

The realization and quasi realization problem for hidden Markov models have been posed for the first time in [20]. The *realization problem* for hidden Markov models consists in finding a hidden Markov model with given string probabilities of all finite length output strings. This question can be split up into three different parts. The first part is the *realizability question*: under which conditions are given output string probabilities representable by a hidden Markov model. The second part is the *realization problem* itself: given realizable string probabilities, find a corresponding minimal hidden Markov model. The last question is the *equivalence problem*: given relizable string probabilities, derive *all* corresponding hidden Markov models.

Linear time-invariant stochastic models in state-space form on the other hand have been introduced into the literature in [42, 43]. However, it was undoubtedly Kalman who brought the state-space point of view into center stage in system theory [61,62,64]. The realization problem for linear stochastic models was studied in [3, 46, 51]. The realization problem consists of finding a linear stochastic model corresponding to a given autocovariance sequence. Again, the realization problem consists of three parts: the realizability question, the realization problem itself and the equivalence problem.

In this chapter, we formally introduce hidden Markov models and linear stochastic models. We also consider the equivalence problem for Moore and Mealy hidden Markov models. The equivalence problem for hidden Markov models of the deterministic function of a Markov chain type has already been considered in [58]. The equivalence problem for hidden Markov models is compared with the equivalence problem for linear stochastic models. The other subproblems of the realization problem are considered in the next two chapters.

List of own contributions

We here describe our contributions to the equivalence problem for hidden Markov models.

- In Section 3.2.4.1 we describe a procedure to test whether a given quasi Mealy hidden Markov model is minimal and in Section 3.2.4.2 we describe a procedure to find a minimal quasi Mealy model that is equivalent to a given nonminimal quasi or positive hidden Markov model.

- In Section 3.3 we consider the equivalence problem for hidden Markov models. We make a distinction between Moore and Mealy hidden Markov models. In Section 3.3.1.2 we provide a test to check whether two positive Mealy hidden Markov models are equivalent and give a description of the complete set of equivalent models. We make a distinction between positive Mealy models that are minimal as a quasi model and positive Mealy models that are not minimal as a quasi Mealy model. Subsequently, we prove that Moore models that are minimal as a quasi Mealy model, under certain conditions, do not have non-trivial equivalents. We show that Moore models that are not minimal as a quasi model can have non-trivial equivalents. In Section 3.3.2 we provide a test for checking the equivalence of Moore models and give a description of the complete set of equivalent models.

- In Section 3.5 we prove that a Moore linear stochastic model which is minimal as a Mealy model, under certain conditions, has only trivial equivalents. We show that the equivalence problem for hidden Markov models is completely analogous to the equivalence problem for linear stochastic models.

Section-by-section overview

In Section 3.1 we introduce finite-valued processes. In Section 3.2 we introduce two types of hidden Markov models that can be used to model finite-valued processes: Moore and Mealy hidden Markov models. We also describe the conversion between both types of models and minimality of the models. In Section 3.3 we consider the equivalence problem for hidden Markov models. We make a distinction between Mealy and Moore models. In Section 3.4 we formally define linear stochastic models and in Section 3.5 we consider the equivalence problem for linear stochastic models.

3.1 Finite valued processes

A *(finite valued) process* y is defined on the time axis $\mathbb{N}$ and takes values in a finite set $\mathbb{Y}$. The set $\mathbb{Y}$ with $|\mathbb{Y}| < \infty$ is called the *output set.*

Denote by $\mathbb{Y}^t$ the set of all strings of length t with symbols from the set $\mathbb{Y}$, and by $\mathbb{Y}^*$ the set of all finite length strings including the empty string ϕ. With $\mathbf{u} = u_1 u_2 \dots u_{|\mathbf{u}|}$ we denote a string from $\mathbb{Y}^*$, where $|\mathbf{u}|$ is the length of $\mathbf{u}$.

The process y is called *stationary* if

$$P(y(t) = u_1, y(t+1) = u_2, \dots, \mathrm{y}(t + |\mathbf{u}| - 1) = u_{|\mathbf{u}|}),$$

is independent of t for all $\mathbf{u} \in \mathbb{Y}^*$.

The process y is completely described by its *string probabilities* $\mathcal{P}_y : \mathbb{Y}^* \mapsto [0, 1]$, defined by

$$\mathcal{P}_y(\mathbf{u}) := P(y(1) = u_1, y(2) = u_2, \dots, y(|\mathbf{u}|) = u_{|\mathbf{u}|}). \tag{3.1}$$

String probabilities $\mathcal{P}_y$ satisfy the following *consistency conditions*

$$\begin{aligned} \mathcal{P}_y(\phi) &= 1, \\ \sum_{\mathrm{y} \in \mathbb{Y}} \mathcal{P}_y(\mathbf{u}\mathrm{y}) &= \mathcal{P}_y(\mathbf{u}), \quad \forall\ \mathbf{u} \in \mathbb{Y}^*, \\ \sum_{\mathbf{u} \in \mathbb{Y}^k} \mathcal{P}_y(\mathbf{u}) &= 1, \quad \forall\ k \in \mathbb{N}. \end{aligned}$$

If the process y is stationary, then in addition,

$$\sum_{\mathrm{y} \in \mathbb{Y}} \mathcal{P}_y(\mathrm{y}\mathbf{u}) = \mathcal{P}_y(\mathbf{u}), \quad \forall\ \mathbf{u} \in \mathbb{Y}^*,$$

so that we have for stationary processes that $\sum_{\mathrm{y} \in \mathbb{Y}} \mathcal{P}(\mathbf{u}\mathrm{y}) = \sum_{\mathrm{y} \in \mathbb{Y}} \mathcal{P}(\mathrm{y}\mathbf{u})$, for all $\mathbf{u} \in \mathbb{Y}^*$.

Suppose $\mathbf{y} : \mathbb{N} \mapsto \mathbb{Y}$ is a realization of the process y. Then the *sample string probabilities* $\mathcal{R}_{\mathbf{y}} : \mathbb{Y}^* \mapsto [0, 1]$ of the realization $\mathbf{y}$ are defined as

$$\mathcal{R}_{\mathbf{y}}(\mathbf{u}) := \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} \delta\left(\mathbf{y}(k)\mathbf{y}(k+1) \dots \mathbf{y}(|\mathbf{u}| + k - 1), \mathbf{u}\right), \tag{3.2}$$

where $\delta(\cdot, \cdot)$ is the Kronecker delta. The string probabilities (3.1) are defined as an ensemble average, but the sample string probabilities (3.2) are defined as a time average. In data analysis applications, we deal with the sample of a process, generated from a particular experiment rather than an ensemble. Hence, in data analysis applications, definition (3.2) is preferable over definition (3.1). However for ergodic processes, it holds that $\mathcal{P}_y = \mathcal{R}_{\mathbf{y}}$. An ergodic process is a stationary process whose statistical properties are determined from its sample process. In this thesis, we only consider ergodic processes.

However, data analysis problems of ergodic processes typically involve a sequence $\mathbf{y}^{(T)}$ of length T with $T < \infty$ instead of an infinite sample process $\mathbf{y}$. Therefore, only *approximate string probabilities* $\tilde{\mathcal{P}}_{\mathbf{y}}^{(t)} : \{\mathbf{u} \in \mathbb{Y}^* : |\mathbf{u}| \leq t\} \mapsto [0,1]$ with $t \leq T$ can be calculated

$$\begin{aligned}
\tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathbf{u}) &:= \frac{1}{T-|\mathbf{u}|+1} \sum_{k=1}^{T-|\mathbf{u}|+1} \delta\left(\mathbf{y}(k)\mathbf{y}(k+1)\dots\mathbf{y}(|\mathbf{u}|+k-1), \mathbf{u}\right), \quad && |\mathbf{u}| = t, \\
\tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathbf{u}) &:= \sum_{\mathrm{y}\in\mathbb{Y}} \tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathbf{u}\mathrm{y}), && |\mathbf{u}| < t.
\end{aligned} \tag{3.3}$$

Approximate string probabilities $\tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathbf{u})$ of a sample process $\mathbf{y}$ satisfy the following *consistency conditions*

$$\begin{aligned}
\tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\phi) &= 1, \\
\sum_{\mathrm{y}\in\mathbb{Y}} \tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathbf{u}\mathrm{y}) &= \tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathbf{u}), && \forall\ \mathbf{u} \in \mathbb{Y}^*,\ |\mathbf{u}| \leq t-1, \\
\sum_{\mathbf{u}\in\mathbb{Y}^k} \tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathbf{u}) &= 1, && \forall\ k \in \mathbb{N},\ k \leq t.
\end{aligned}$$

Approximate string probabilities $\tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathbf{u})$ of a sample process $\mathbf{y}$ are called *stationary* if they satisfy

$$\sum_{\mathrm{y}\in\mathbb{Y}} \tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathrm{y}\mathbf{u}) = \tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathbf{u}), \quad \forall\ \mathbf{u} \in \mathbb{Y}^*,\ |\mathbf{u}| \leq t-1. \tag{3.4}$$

If the approximate string probabilities are stationary, it holds that $\sum_{\mathrm{y}\in\mathbb{Y}} \tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathbf{u}\mathrm{y}) = \sum_{\mathrm{y}\in\mathbb{Y}} \tilde{\mathcal{P}}_{\mathbf{y}}^{(t)}(\mathrm{y}\mathbf{u})$, for all $\mathbf{u} \in \mathbb{Y}^*,\ |\mathbf{u}| \leq t-1$.

Example 3.1. *Consider a sequence $\mathbf{y}^{(T)}$ with length $T = 20$ taking values in $\mathbb{Y} = \{0,1\}$*

$$\mathbf{y}^{(T)} = 01010100110011010001.$$

The approximate string probabilities $\tilde{\mathcal{P}}_{\mathbf{y}}^{(3)}$ of strings of length 3 can be obtained by counting the number of occurences of a certain string in the sequence $\mathbf{y}^{(T)}$ divided by the number of times that the string could have occured:

$$\begin{aligned}
\tilde{\mathcal{P}}_{\mathbf{y}}^{(3)}(000) &= \frac{1}{18}, & \tilde{\mathcal{P}}_{\mathbf{y}}^{(3)}(100) &= \frac{3}{18}, \\
\tilde{\mathcal{P}}_{\mathbf{y}}^{(3)}(001) &= \frac{3}{18}, & \tilde{\mathcal{P}}_{\mathbf{y}}^{(3)}(101) &= \frac{3}{18}, \\
\tilde{\mathcal{P}}_{\mathbf{y}}^{(3)}(010) &= \frac{4}{18}, & \tilde{\mathcal{P}}_{\mathbf{y}}^{(3)}(110) &= \frac{2}{18}, \\
\tilde{\mathcal{P}}_{\mathbf{y}}^{(3)}(011) &= \frac{2}{18}, & \tilde{\mathcal{P}}_{\mathbf{y}}^{(3)}(111) &= 0,
\end{aligned}$$

The approximate string probabilities $\tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}$ of strings of length smaller than 3*, can be obtained from the string probabilities of strings of length* 3*:*

$$\begin{aligned}
\tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(00) &= \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(000) + \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(001) = \tfrac{4}{18},\\
\tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(01) &= \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(010) + \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(011) = \tfrac{6}{18},\\
\tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(10) &= \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(100) + \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(101) = \tfrac{6}{18},\\
\tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(11) &= \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(110) + \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(111) = \tfrac{2}{18},
\end{aligned}$$

and

$$\begin{aligned}
\tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(0) &= \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(00) + \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(01) = \tfrac{10}{18},\\
\tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(1) &= \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(10) + \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(11) = \tfrac{8}{18},
\end{aligned}$$

and

$$\tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(\phi) = \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(0) + \tilde{\mathcal{P}}^{(3)}_{\mathbf{y}}(1) = 1.$$

The approximate string probabilities are consistent (by construction). However, it is easy to verify that condition (3.4) does not hold for all $\mathbf{u} \in \mathbb{Y}^*, |\mathbf{u}| \leq 2$. We hence conclude that the approximate string probabilities are not stationary. □

In the rest of the thesis, we use $\mathcal{P}$, $\mathcal{R}$ and $\tilde{\mathcal{P}}^{(t)}$ instead of $\mathcal{P}_y$, $\mathcal{R}_{\mathbf{y}}$ and $\tilde{\mathcal{P}}^{(t)}_{\mathbf{y}}$, when the process or sample process to which the string probabilities belong is clear from the context.

3.2 Hidden Markov models

In the previous section we defined finite valued processes. In this section we describe two types of hidden Markov models that can be used to model this kind of processes.

In Section 3.2.1 we describe Mealy hidden Markov models and in Section 3.2.2 Moore hidden Markov models. In Section 3.2.3 we explain how to convert a Mealy in a Moore model and vice versa. Section 3.2.4 deals with the minimality of hidden Markov models.

3.2.1 Mealy hidden Markov models

The finite-valued process y taking values in $\mathbb{Y}$ with $|\mathbb{Y}| < \infty$, is said to be *representable by a Mealy hidden Markov model (Mealy HMM)* if there exists a finite-valued process x taking values in $\mathbb{X}$ with $|\mathbb{X}| < \infty$ such that

$$P\left(\begin{array}{c} x(p_m) = \mathtt{x}^{(p_m)}, \ldots, x(p_1) = \mathtt{x}^{(p_1)}, x(f_1) = \mathtt{x}^{(f_1)}, \ldots, x(f_n) = \mathtt{x}^{(f_n)} \\ y(p_m) = \mathtt{y}^{(p_m)}, \ldots, y(p_1) = \mathtt{y}^{(p_1)}, y(f_1) = \mathtt{y}^{(f_1)}, \ldots, y(f_n) = \mathtt{y}^{(f_n)} \end{array} \middle| x(f_0) \right)$$
$$= P\left(\begin{array}{c} x(p_m) = \mathtt{x}^{(p_m)}, \ldots, x(p_1) = \mathtt{x}^{(p_1)} \\ y(p_m) = \mathtt{y}^{(p_m)}, \ldots, y(p_1) = \mathtt{y}^{(p_1)} \end{array} \middle| x(f_0) \right) \cdot$$
$$P\left(\begin{array}{c} x(f_1) = \mathtt{x}^{(f_1)}, \ldots, x(f_n) = \mathtt{x}^{(f_n)} \\ y(f_1) = \mathtt{y}^{(f_1)}, \ldots, y(f_n) = \mathtt{y}^{(f_n)} \end{array} \middle| x(f_0) \right), \quad (3.5)$$

for all $p_1, \ldots, p_m, f_0, \ldots, f_n \in \mathbb{N}$ with $p_m \leq \ldots \leq p_1 \leq f_0 \leq \ldots \leq f_n$, for all $\mathtt{x}^{(p_m)}, \ldots, \mathtt{x}^{(p_1)}, \mathtt{x}^{(f_0)}, \ldots, \mathtt{x}^{(f_n)} \in \mathbb{X}$ and for all $\mathtt{y}^{(p_m)}, \ldots, \mathtt{y}^{(p_1)}, \mathtt{y}^{(f_0)}, \ldots, \mathtt{y}^{(f_n)} \in \mathbb{Y}$. The process x is called the *state process* with $\mathbb{X}$ the *state set* and the process y is called the *output process* with $\mathbb{Y}$ the *output set*. Without loss of generality, we identify $\mathbb{X} = \{1, 2, \ldots, |\mathbb{X}|\}$. In words, (3.5) says that the past of $\left[\begin{array}{c} x \\ y \end{array} \right]$ is independent of its present and future given the present state. So, to compute the future of a process from the past of a process it is enough to have the present state. This is an important property of the state: it summarizes all the important information of the past needed to compute the future.

It follows from (3.5) that x is a *Markov process*, i.e.

$$P(x(f_1) = \mathtt{x}^{(f_1)}, \ldots, x(f_n) = \mathtt{x}^{(f_n)} | x(p_m), \ldots, x(p_1), x(f_0)) =$$
$$P(x(f_1) = \mathtt{x}^{(f_1)}, \ldots, x(f_n) = \mathtt{x}^{(f_n)} | x(f_0)),$$

and that y is a *probabilistic function of the Markov process* x, which means that

$$P\left(y(f_0) = \mathtt{y}^{(f_0)}, \ldots, y(f_n) = \mathtt{y}^{(f_n)} \middle| \begin{array}{c} x(p_m), \ldots, x(p_1), x(f_0) \\ y(p_m), \ldots, y(p_2), y(p_1) \end{array} \right) =$$
$$P(y(f_0) = \mathtt{y}^{(f_0)}, \ldots, y(f_n) = \mathtt{y}^{(f_n)} | x(f_0)).$$

If $P(x(t+1), y(t)|x(t))$ is independent of t (i.e. the joint process $\left[\begin{array}{c} x \\ y \end{array} \right]$ is time-homogeneous), then the process is representable by a *time-homogeneous (positive) Mealy hidden Markov model*[1] defined as $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ where:

- Π is a mapping from $\mathbb{Y}$ to $\mathbb{R}_+^{|\mathbb{X}| \times |\mathbb{X}|}$, with $\Pi_{\mathbb{X}} := \sum_{\mathtt{y} \in \mathbb{Y}} \Pi(\mathtt{y})$ a stochastic matrix, i.e. $\Pi_{\mathbb{X}} e = e$. The element $\Pi_{ij}(\mathtt{y})$ is $P(x(t+1) = j, y(t) = \mathtt{y} | x(t) = i)$, that is, the probability of going from state i to state j while generating output symbol $\mathtt{y}$. The matrix $\Pi_{\mathbb{X}}$ is called the *state transition matrix* of the HMM.

[1] When refering to a Mealy hidden Markov model, the word "positive" may be added to make a distinction with a "quasi" hidden Markov model (defined further). If "positive" or "quasi" is omitted, it should be clear from the context, which one of both models is mentioned.

- $\pi(1)$ is a vector in $\mathbb{R}_+^{1\times|\mathbb{X}|}$ for which $\pi(1)e = 1$. It is called the *initial state distribution.* The element $\pi_i(1)$ is $P(x(1) = i)$, that is the probability that the initial state is i.

In the remainder of the thesis, we consider only hidden Markov models that are time-homogeneous. So we omit the word "time-homogeneous" when refering to a time-homogeneous (positive) Mealy hidden Markov model.

The number of states $|\mathbb{X}|$ is called the *order* of the HMM. The conditions $\pi(1)e = 1$ and $\Pi_{\mathbb{X}}e = e$ are called *consistency conditions* of the HMM. If the initial state distribution vector is a left eigenvector of the state transition matrix corresponding to the eigenvalue 1: $\pi(1)\Pi_{\mathbb{X}} = \pi(1)$, the state process is stationary. As a consequence the output process y is also stationary and the HMM is called *stationary.* The string probabilities *generated* by a Mealy HMM $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ are given by

$$\mathcal{P}(\mathbf{u}) \quad = \quad \pi(1)\Pi(\mathbf{u})e,$$

where $\mathbf{u} = u_1u_2\dots u_{|\mathbf{u}|} \in \mathbb{Y}^*$ and where $\Pi(\mathbf{u}) := \Pi(u_1)\Pi(u_2)\dots\Pi(u_{|\mathbf{u}|})$.

Example 3.2. *Consider the Mealy model* $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ *with* $\mathbb{X} = \{1, 2, 3, 4\}$, $\mathbb{Y} = \{a, b, c, d\}$,

$$\Pi(a) = \begin{bmatrix} 0 & 0.09 & 0 & 0 \\ 0 & 0.09 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0.01 & 0 \end{bmatrix}, \quad \Pi(b) = \begin{bmatrix} 0 & 0.01 & 0 & 0 \\ 0 & 0.01 & 0 & 0.81 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.81 \end{bmatrix},$$

$$\Pi(c) = \begin{bmatrix} 0.81 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.09 \\ 0.81 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.09 \end{bmatrix}, \quad \Pi(d) = \begin{bmatrix} 0.09 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.09 & 0 & 0.09 & 0 \\ 0 & 0 & 0.09 & 0 \end{bmatrix},$$

and

$$\pi(1) = \begin{bmatrix} 0.45 & 0.05 & 0.05 & 0.45 \end{bmatrix}.$$

All elements in the matrices $\Pi(a)$, $\Pi(b)$, $\Pi(c)$, $\Pi(d)$ *and* $\pi(1)$ *represent probabilities. For instance,* $\Pi_{1,2}(a) = 0.09$ *means*

$$\Pi_{1,2}(a) = P(y(t) = a, x(t + 1) = 2|x(t) = 1) = 0.09,$$

and $\pi_2(1) = 0.05$ *means*

$$\pi_2(1) = P(x(1) = 2) = 0.05.$$

Note that the model is stationary as $\pi(1)$ *is a left eigenvector of* $\sum_{y\in\mathbb{Y}} \Pi(y)$.

We now calculate the string probabilities generated by the hidden Markov model for the string "aba" and the string "dd":

$$\begin{aligned} \mathcal{P}(aba) &= \pi(1)\Pi(a)\Pi(b)\Pi(a)e = 0.000405, \\ \mathcal{P}(dd) &= \pi(1)\Pi(d)\Pi(d)e = 0.0121. \end{aligned}$$

□

We can now formulate the Mealy realization problem and the minimal Mealy realization problem[2].

Problem 3.1 (Mealy realization problem)**.** *Given output string probabilities* $\mathcal{P} : \mathbb{Y}^* \mapsto [0,1]$*. Find a Mealy HMM* $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ *that generates* $\mathcal{P}$*.*

Problem 3.2 (minimal Mealy realization problem)**.** *Given output string probabilities* $\mathcal{P} : \mathbb{Y}^* \mapsto [0,1]$*. Find a Mealy HMM* $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$*, with* $|\mathbb{X}|$ *as small as possible, that generates* $\mathcal{P}$*.*

It is clear that the minimal realization problem is the most interesting. An implicit subproblem of the Mealy realization problem is the *Mealy realizability question*: derive conditions for string probabilities $\mathcal{P}$ to be representable by a Mealy HMM of finite order. A solution to the Mealy realization problem is called a *realization* of the string probabilities $\mathcal{P}$. The realization problem is hard because of the positivity contraints on $\pi(1)$ and Π. For that reason, we consider a relaxed version of the problem called the quasi realization problem. The quasi realization problem is the same problem as the realization problem but without the positivity contraints. The quasi model which is found from the quasi realization procedure retains important properties of a positive model. In several applications it suffices to have a quasi instead of a positive hidden Markov model (see Chapter 7).

Note that the word "realization" has two different meanings. As explained before, a realization $\mathbf{y}$ of a process y is one sample sequence of the process. On the other hand, a realization of the string probabilities $\mathcal{P}$ is a solution to the realization problem. In the rest of the text, it should be clear from the context which one is mentioned.

A *quasi (Mealy) HMM* is defined by $(\mathbb{Q}, \mathbb{Y}, A, c, b)$, where $\mathbb{Q}$ is the *quasi state set* and $\mathbb{Y}$ is the *output set*. The number of states $|\mathbb{Q}|$ is called the *order* of the quasi HMM. b is a column vector in $\mathbb{R}^{|\mathbb{Q}|}$, A is a mapping from $\mathbb{Y}$ to $\mathbb{R}^{|\mathbb{Q}|\times|\mathbb{Q}|}$, where $A_{\mathbb{Q}} := \sum_{\mathrm{y}\in\mathbb{Y}} A(\mathrm{y})$ is a quasi stochastic matrix, i.e. $A_{\mathbb{Q}}b = b$. The matrices A, c, b are such that $cA(\mathbf{u})b \in [0,1]$ for all $\mathbf{u} = u_1 u_2 \dots u_{|\mathbf{u}|} \in \mathbb{Y}^*$, where $A(\mathbf{u}) := A(u_1)A(u_2)\dots A(u_{|\mathbf{u}|})$. The matrix $A_{\mathbb{Q}}$ is called the *quasi state transition matrix*. c is a vector in $\mathbb{R}^{1\times|\mathbb{Q}|}$ called the *quasi initial state distribution* for which $cb = 1$. The conditions $cb = 1$ and $A_{\mathbb{Q}}b = b$ are called *consistency conditions* of the quasi HMM. The quasi HMM is called *stationary* if the quasi initial state distribution vector is a left eigenvector of the quasi state transition matrix corresponding to the eigenvalue 1: $cA = c$. Notice that A, c and b of a quasi Mealy model $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ are the analogues of Π, $\pi(1)$ and e of a positive Mealy model $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$.

The string probabilities $\mathcal{P} : \mathbb{Y}^* \mapsto [0,1]$ *generated by* a quasi Mealy HMM $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ are given by

$$\mathcal{P}(\mathbf{u}) \;=\; cA(\mathbf{u})b,$$

[2]In this thesis, we consider the "weak realization problem". The weak realization problem aims at modeling the statistics of a process (string probabilities, autocovariances). In the remainder of the thesis, the word "weak" is omitted when referring to the weak realization problem.

where $\mathbf{u} = u_1 u_2 \ldots u_{|\mathbf{u}|} \in \mathbb{Y}^*$ and where $A(\mathbf{u}) := A(u_1)A(u_2)\ldots A(u_{|\mathbf{u}|})$. Notice that a Mealy HMM is also a quasi Mealy HMM, but a quasi Mealy HMM is not necessary a Mealy HMM.

We now formulate the quasi Mealy realization problem.

Problem 3.3 (quasi Mealy realization problem)**.** *Given output string probabilities $\mathcal{P} : \mathbb{Y}^* \mapsto [0,1]$. Find a quasi Mealy HMM $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ that generates $\mathcal{P}$.*

Problem 3.4 (minimal quasi Mealy realization problem)**.** *Given output string probabilities $\mathcal{P} : \mathbb{Y}^* \mapsto [0,1]$. Find a quasi Mealy HMM $(\mathbb{Q}, \mathbb{Y}, A, c, b)$, with $|\mathbb{Q}|$ as small as possible, that generates $\mathcal{P}$.*

Again, an implicit subproblem of the quasi Mealy realization problem is the *quasi Mealy realizability question*: derive conditions for string probabilities $\mathcal{P}$ to be representable by a quasi Mealy HMM of finite order. A solution to the quasi Mealy realization problem is called a *quasi realization* of the string probabilities $\mathcal{P}$. We now define equivalence and minimality of hidden Markov models.

Definition 3.1. *Two Mealy HMMs (either both positive, both quasi or one positive and one quasi) with string probabilities $\mathcal{P}$ and $\mathcal{P}'$ respectively, are said to be* equivalent *if $\mathcal{P} = \mathcal{P}'$.*

Definition 3.2. *A Mealy HMM $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ is called* minimal *if for any other equivalent Mealy model $(\mathbb{X}', \mathbb{Y}, \Pi', \pi'(1))$ it holds that $|\mathbb{X}| \leq |\mathbb{X}'|$.*

Definition 3.3. *A quasi Mealy model $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ is called* minimal *if for any other equivalent quasi Mealy model $(\mathbb{Q}', \mathbb{Y}, A', c', b')$ it holds that $|\mathbb{Q}| \leq |\mathbb{Q}'|$.*

Definition 3.4. *A Mealy HMM $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ is called* minimal as a quasi Mealy model *if for any other equivalent quasi Mealy model $(\mathbb{Q}', \mathbb{Y}, A', c', b')$ it holds that $|\mathbb{X}| \leq |\mathbb{Q}'|$.*

It is clear that the order of a minimal quasi HMM is smaller than or equal to the order of a minimal equivalent positive HMM. It follows from the definitions above that a solution to the minimal Mealy realization problem is always a minimal Mealy model. Also, a solution to the minimal quasi Mealy models is a minimal quasi Mealy model. On the other hand, a solution of the minimal Mealy realization problem can be obtained by first calculating a solution to the Mealy realization problem and subsequently computing an equivalent minimal Mealy model. The same holds for the minimal quasi Mealy realization problem.

Example 3.3. *Consider the quasi Mealy model given by $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ with $\mathbb{Q} = \{1,2,3\}$, $\mathbb{Y} = \{a,b,c,d\}$,*

$$A(a) = \begin{bmatrix} 0.000 & 0.000 & 0.000 \\ -0.050 & 0.050 & -0.040 \\ 0.040 & -0.040 & 0.050 \end{bmatrix}, \quad A(b) = \begin{bmatrix} 0.405 & 0.405 & 0.405 \\ -0.005 & 0.005 & -0.005 \\ 0.410 & 0.400 & 0.410 \end{bmatrix},$$

$$A(c) = \begin{bmatrix} 0.450 & 0.450 & -0.360 \\ 0.000 & 0.000 & 0.000 \\ -0.360 & -0.360 & 0.450 \end{bmatrix}, \quad A(d) = \begin{bmatrix} 0.045 & 0.045 & -0.045 \\ -0.045 & 0.045 & 0.045 \\ -0.090 & 0.000 & 0.060 \end{bmatrix},$$

$$c = \begin{bmatrix} 0.000 & 0.000 & -0.500 \end{bmatrix},$$

$$b = \begin{bmatrix} 0.000 & 0.000 & 2.000 \end{bmatrix}^\top .$$

Some of the elements in the matrices $A(a)$, $A(b)$, $A(c)$ and $A(d)$ are negative and hence the elements of A do not represent probabilities. However, the string probabilities generated by the hidden Markov models are nonnegative. We compute the string probabilities generated by the quasi hidden Markov model for the string "aba" and the string "dd":

$$\begin{aligned} \mathcal{P}(aba) &= cA(a)A(b)A(a)b = 0.000405, \\ \mathcal{P}(dd) &= cA(d)A(d)b = 0.0121. \end{aligned}$$

The string probabilities for the strings "aba" and "dd" generated by the quasi Mealy model $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ are exactly equal to the string probabilities generated by the positive Mealy model $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ of Example 3.2. It can be shown that this equality holds for all strings, from which we conclude that the quasi Mealy model $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ and the Mealy model $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ are equivalent (see Example 3.4). Moreover, both the quasi model $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ as the positive model $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ are minimal. In this example, the order of a minimal quasi HMM is smaller than the order of a minimal equivalent positive HMM. □

3.2.2 Moore hidden Markov models

The process y taking values in $\mathbb{Y}$ with $|\mathbb{Y}| < \infty$, is said to be *representable by a Moore hidden Markov model (Moore HMM)* if there exists another process x taking values in $\mathbb{X}$ with $|\mathbb{X}| < \infty$ such that (3.5) holds and such that

$$P(x(t+1), y(t)|x(t)) = P(x(t+1)|x(t)) \cdot P(y(t)|x(t)),$$

for all $t \in \mathbb{N}$. The process x is called the *state process* and the process y is called the *output process*. Without loss of generality, we identify $\mathbb{X} = \{1, 2, \ldots, |\mathbb{X}|\}$. It follows from (3.5) that x is a *Markov process*, and that y is a *probabilistic function of the Markov process x*.

If $P(x(t+1), y(t)|x(t))$ is independent of t (i.e. the joint process $\begin{bmatrix} x \\ y \end{bmatrix}$ is time-homogeneous), then the process is representable by a *time-homogeneous Moore hidden Markov model* defined as $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, \beta, \pi(1))$ where $\mathbb{X}$ and $\mathbb{Y}$ are the *state* and *output sets* and where

- $\Pi_{\mathbb{X}} \in \mathbb{R}_+^{|\mathbb{X}| \times |\mathbb{X}|}$ with $\Pi_{\mathbb{X}} e = e$ is the *state transition matrix*, defined as $(\Pi_{\mathbb{X}})_{ij} = P(x(t+1) = j | x(t) = i)$.
- β is a mapping from $\mathbb{Y}$ to $\mathbb{R}_+^{|\mathbb{X}|}$, defined as $\beta_i(\mathtt{y}) = P(y(t) = \mathtt{y} | x(t) = i)$ and is called the *output map*. It is required that $\sum_{\mathtt{y}} \beta(\mathtt{y}) = e$.
- $\pi(1)$ is a vector in $\mathbb{R}_+^{1 \times |\mathbb{X}|}$ defined as $\pi_i(1) = P(x(1) = i)$ for which $\pi(1)e = 1$. It is called the *initial state distribution*.

In the remainder of the thesis, we consider only hidden Markov models that are time-homogeneous such that we omit the word "time-homogeneous" when refering to a time-homogeneous Moore hidden Markov model.

The number of states $|\mathbb{X}|$ is the *order* of the Moore model. Suppose we have an ordering $(\mathsf{y}_k, k = 1, 2, \dots, |\mathbb{Y}|)$ of the symbols of the output set $\mathbb{Y}$, then the output map β can be represented by a matrix, called the *output matrix* B defined as $B := \left[\begin{array}{ccc} \beta(\mathsf{y}_1) & \dots & \beta(\mathsf{y}_{|\mathbb{Y}|}) \end{array}\right]$, with $Be = e$. An equivalent description of the Moore HMM is therefore given by $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, B, \pi(1))$. The conditions $\Pi_{\mathbb{X}} e = e$, $\sum_{\mathsf{y}} \beta(\mathsf{y}) = e$ or $Be = e$ and $\pi(1)e = 1$ are called *consistency conditions* of the HMM. A Moore HMM is called *stationary* if $\pi(1)\Pi_{\mathbb{X}} = \pi(1)$. String probabilities *generated by* a Moore HMM $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, \beta, \pi(1))$ are given by

$$\mathcal{P}(\mathrm{y}) \quad = \quad \pi(1) \operatorname{diag}(\beta(\mathrm{y}_1))\Pi_{\mathbb{X}} \dots \operatorname{diag}(\beta(\mathrm{y}_{|\mathrm{y}|}))\Pi_{\mathbb{X}} e,$$

where $\mathrm{y} = \mathrm{y}_1\mathrm{y}_2 \dots \mathrm{y}_{|\mathrm{y}|} \in \mathbb{Y}^*$.

We can now formulate the Moore realization problem and the minimal Moore realization problem.

Problem 3.5 (Moore realization problem)**.** *Given output string probabilities $\mathcal{P} : \mathbb{Y}^* \mapsto [0, 1]$. Find a Moore HMM $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, \beta, \pi(1))$ that generates $\mathcal{P}$.*

Problem 3.6 (minimal Moore realization problem)**.** *Given output string probabilities $\mathcal{P} : \mathbb{Y}^* \mapsto [0, 1]$. Find a minimal Moore HMM $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, \beta, \pi(1))$, with $|\mathbb{X}|$ as small as possible, that generates $\mathcal{P}$.*

An implicit subproblem of the Moore realization problem is the *Moore realizability question*: derive conditions for string probabilities $\mathcal{P}$ to be representable by a Moore HMM of finite order. A solution to the Moore realization problem is called a *Moore realization* of the string probabilities $\mathcal{P}$. Analogous to the situation for Mealy hidden Markov models, one could define quasi Moore models and the corresponding quasi Moore realization problem. However, working with quasi Moore models does not give much advantage. For that reason, we do not go into detail about quasi Moore models.

We now give the most complete definition of equivalence of hidden Markov models.

Definition 3.5. *Two HMMs (both either positive Mealy, either quasi Mealy or Moore) with string probabilities $\mathcal{P}$ and $\mathcal{P}'$ respectively, are said to be* equivalent *if $\mathcal{P} = \mathcal{P}'$.*

As for quasi Mealy HMMs, we define two types of minimality for Moore HMMs.

Definition 3.6. *A Moore HMM $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, \beta, \pi(1))$ is called* minimal *if for any other equivalent Moore model $(\mathbb{X}', \mathbb{Y}, \Pi'_{\mathbb{X}}, \beta', \pi'(1))$ it holds that $|\mathbb{X}| \leq |\mathbb{X}'|$.*

Definition 3.7. *A Moore HMM $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, \beta, \pi(1))$ is called* minimal as a quasi Mealy model *if for any other equivalent quasi Mealy model $(\mathbb{Q}', \mathbb{Y}, A', c', b')$ it holds that $|\mathbb{X}| \leq |\mathbb{Q}'|$.*

It follows from the definitions above that a solution to the minimal Moore realization problem is always a minimal Moore model. On the other hand, a solution of the minimal Moore realization problem can be obtained by first calculating a solution to the Moore realization problem and subsequently calculating a minimal equivalent Moore model.

3.2.3 Mealy versus Moore HMMs

Mealy and Moore hidden Markov models are different models. The difference is due to the way output symbols are generated. For a Mealy model the event of producing an output symbol at the present time instant given the present state and the event of going to a next state given the present state are dependent. For a Moore model these events are independent. In Figure 3.1, the difference between Mealy and Moore hidden Markov models is schematically shown. However, it can be shown that the "expressive power" of Moore HMMs and Mealy HMMs is the same [112, 113], meaning that the Mealy realization problem of string probabilities $\mathcal{P}$ has a solution if and only if the Moore realization problem of the same string probabilities $\mathcal{P}$ has a solution. It is clear that for given string probabilities the order of a minimal Mealy model of string probabilities does not exceed the order of a minimal Moore model.

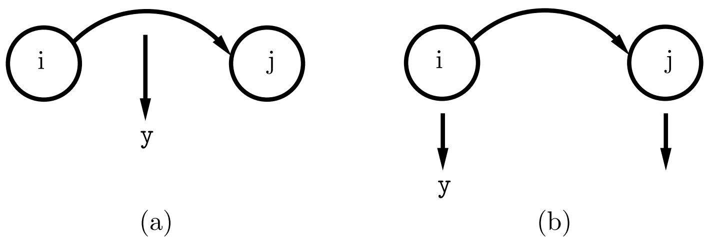


Figure 3.1: *For a Mealy model (Subfigure (a)) the event of producing an output symbol and the event of going to a next state are dependent. For a Moore model (Subfigure (b)) these events are independent.*

Converting a Moore hidden Markov model $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, \beta, \pi(1))$ into a Mealy hidden Markov model $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ is always possible, using

$$\Pi(\mathtt{y}) \quad = \quad \mathrm{diag}(\beta(\mathtt{y}))\Pi_{\mathbb{X}}.$$

The obtained Mealy model can be nonminimal, even if the Moore model is minimal.

Converting a Mealy model in a Moore model is also always possible. A Mealy model is described by an output distribution for every state transition, while a Moore model is described by an output distribution for every state. Now, given a Mealy model, one can construct a Moore model by connecting a state of the Moore model to every state transition of the Mealy model, and then calculating the state transition probabilities and output probabilities in the appropriate way. Typically, this approach leads to a highly nonminimal Moore model even if the Mealy model is minimal.

3.2.4 Minimality of HMMs

For positive HMMs, to the best of our knowledge, there does not exist any test for minimality and no procedure to obtain a minimal model equivalent to a given nonminimal model (open problem on p. 54 of [81]). In this section we consider minimality of quasi Mealy models.

In Section 3.2.4.1 we explain a test for checking whether a quasi Mealy model is minimal. In Section 3.2.4.2 subsequently, we describe a procedure to find a minimal quasi Mealy model that is equivalent to a positive Mealy model or to a nonminimal quasi Mealy model.

3.2.4.1 Test for minimality of quasi Mealy HMMs

An ordered set of strings $\mathcal{M} := (\mathbf{m}_k, k = 1, 2, \ldots)$ is said to be in *first lexicographical ordering* if the strings are ordered lexicographically from right to left, and such that the length of the strings $|\mathbf{m}_k|$ increases monotonically with k. In the remainder of the thesis, the symbol $\mathcal{U} := (\mathbf{u}_k, k = 1, 2, \ldots)$ is reserved for the ordered set of all strings of $\mathbb{Y}^*$ in first lexicographical ordering. The first element of $\mathcal{U}$ is hence equal to ϕ. For $\mathbb{Y} = \{0, 1\}$, the ordering $\mathcal{U}$ is given by $(\phi, 0, 1, 00, 10, 01, 11, 000, 100, \ldots)$. On the other hand, an ordered set of strings $\mathcal{N} := (\mathbf{n}_k, k = 1, 2, \ldots)$ is said to be in *last lexicographical ordering* if the strings are ordered lexicographically from left to right, and such that the length of the strings $|\mathbf{n}_k|$ increases monotonically with k. The symbol $\mathcal{V} := (\mathbf{v}_k, k = 1, 2, \ldots)$ is reserved for the ordered set of all strings of $\mathbb{Y}^*$ in last lexicographical ordering. This gives $(\phi, 0, 1, 00, 01, 10, 11, 000, 001, \ldots)$ for $\mathbb{Y} = \{0, 1\}$.

Now, define the $\mathcal{O}(c, A)$-matrix in $\mathbb{R}^{\infty \times |\mathbb{Q}|}$ and the $\mathcal{C}(A, b)$-matrix in $\mathbb{R}^{|\mathbb{Q}| \times \infty}$ of a quasi Mealy HMM $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ as

$$\begin{aligned} \mathcal{O}_{i,:}(c, A) &:= cA(\mathbf{u}_i), && (3.6)\\ \mathcal{C}_{:,j}(A, b) &:= A(\mathbf{v}_j)b, && (3.7) \end{aligned}$$

where $\mathbf{u}_i$ is the i-th element of the set of strings from $\mathbb{Y}^*$ in first lexicographical ordering $\mathcal{U}$ and $\mathbf{v}_j$ is the j-th element of the set of strings from $\mathbb{Y}^*$ in last lexicographical ordering $\mathcal{V}$. In the case where $\mathbb{Y} = \{0, 1\}$ the matrices $\mathcal{O}(c, A)$ and $\mathcal{C}(A, b)$ are

$$\mathcal{O}(c, A) = \left[\begin{array}{c} c \\ \hline cA(0) \\ cA(1) \\ \hline cA(00) \\ cA(10) \\ \vdots \end{array}\right], \qquad \mathcal{C}(A, b) = \left[\begin{array}{c|cc|ccc} b & A(0)b & A(1)b & A(00)b & A(01)b & \ldots \end{array}\right].$$

Let $\mathcal{O}_{(t)}(c,A)$ and $\mathcal{O}_{(1:t)}(c,A)$ for $t \in \mathbb{Z}_+$ be submatrices of $\mathcal{O}(c,A)$ defined as

$$\begin{aligned}\mathcal{O}_{(t)}(c,A) &= [cA(\mathbf{u}_i)], \quad \text{with} \quad |\mathbf{u}_i| = t-1,\\ \mathcal{O}_{(1:t)}(c,A) &= [cA(\mathbf{u}_i)], \quad \text{with} \quad |\mathbf{u}_i| \le t-1.\end{aligned}$$

Analogously, we define the following submatices of $\mathcal{C}(A,b)$

$$\begin{aligned}\mathcal{C}_{(t)}(A,b) &= [A(\mathbf{v}_i)b], \quad \text{with} \quad |\mathbf{v}_i| = t-1,\\ \mathcal{C}_{(1:t)}(A,b) &= [A(\mathbf{v}_i)b], \quad \text{with} \quad |\mathbf{v}_i| \le t-1.\end{aligned}$$

We now prove a proposition that allows us to work with finite versions of the matrices $\mathcal{O}(c,A)$ and $\mathcal{C}(A,b)$.

Proposition 3.1. *Given a quasi Mealy HMM* $(\mathbb{Q},\mathbb{Y},A,c,b)$*. Suppose there exists a scalar* $k \in \mathbb{Z}_+$ *such that*

$$\operatorname{rank}\mathcal{O}_{(1:k)}(c,A) = \operatorname{rank}\mathcal{O}_{(1:k+1)}(c,A),$$

then for $l = 0,1,2,\dots$ *it holds*

$$\operatorname{rank}\mathcal{O}(c,A) = \operatorname{rank}\mathcal{O}_{(1:k+l)}(c,A),$$

Proof: It follows from $\operatorname{rank}\mathcal{O}_{(1:k)}(c,A) = \operatorname{rank}\mathcal{O}_{(1:k+1)}(c,A)$ that there exists a matrix K such that $K\mathcal{O}_{(1:k)}(c,A) = \mathcal{O}_{(k+1)}(c,A)$. It follows that $K\mathcal{O}_{(1:k)}(c,A)A(\mathtt{y}) = \mathcal{O}_{(k+1)}(c,A)A(\mathtt{y})$ for each $\mathtt{y} \in \mathbb{Y}$, from which we conclude that $\operatorname{rank}\mathcal{O}_{(1:k+1)}(c,A) = \operatorname{rank}\mathcal{O}_{(1:k+2)}(c,A)$. By continuing in this way, we find that $\operatorname{rank}\mathcal{O}_{(1:k+1)}(c,A) = \operatorname{rank}\mathcal{O}_{(1:k+2)}(c,A) = \operatorname{rank}\mathcal{O}_{(1:k+3)}(c,A) = \dots$, which proves the proposition. ■

As the rank of $\mathcal{O}(c,A)$ is at most equal to $|\mathbb{Q}|$, the condition of Proposition 3.1 is fullfilled for $k \ge |\mathbb{Q}|$. However, it is possible that there exist a smaller k such that the condition is fullfilled. Thus $\operatorname{rank}\mathcal{O}(c,A) = \operatorname{rank}\mathcal{O}_{(1:|\mathbb{Q}|)}(c,A)$, which is the reason why we frequently use the finite matrix $\mathcal{O}_{(1:|\mathbb{Q}|)}(c,A)$ instead of the infinite matrix $\mathcal{O}(c,A)$. Analogous properties hold for $\mathcal{C}(A,b)$.

We now prove the following theorem, which provides a way to decide about the minimality of a quasi Mealy model.

Theorem 3.1. *The quasi Mealy HMM* $(\mathbb{Q},\mathbb{Y},A,c,b)$ *is minimal if and only if the matrices* $\mathcal{C}_{(1:|Q|)}(A,b)$ *and* $\mathcal{O}_{(1:|Q|)}(c,A)$ *have full row and full column rank respectively.*

Proof: Suppose $(\mathbb{Q},\mathbb{Y},A,c,b)$ is not minimal, then there exists a minimal HMM $(\mathbb{Q}',\mathbb{Y},A',c',b')$ with $|\mathbb{Q}'| < |\mathbb{Q}|$ such that

$$\mathcal{O}_{(1:|Q|)}(c,A)\ \mathcal{C}_{(1:|Q|)}(A,b) = \mathcal{O}_{(1:|Q|)}(c',A')\ \mathcal{C}_{(1:|Q|)}(A',b').$$

Hence the rank of $\mathcal{O}_{(1:|Q|)}(c,A)\ \mathcal{C}_{(1:|Q|)}(A,b)$ is equal to $|\mathbb{Q}'|$, such that it follows from Sylvester's inequality[3] that $\operatorname{rank}\mathcal{O}_{(1:|Q|)}(c,A) + \operatorname{rank}\mathcal{C}_{(1:|Q|)}(A,b) \le |\mathbb{Q}'| +$

[3]Given matrices $A \in \mathbb{R}^{a\times n}$ and $B \in \mathbb{R}^{n\times b}$, then Sylvester's inequality states that $\operatorname{rank}AB \le \operatorname{rank}A + \operatorname{rank}B - n$.

$|\mathbb{Q}|$. It follows that at least one of the matrices $\mathcal{C}_{(1:|Q|)}(A,b)$ and $\mathcal{O}_{(1:|Q|)}(c,A)$ is not of full (row or column) rank.

Now suppose that $\mathcal{C}_{(1:|Q|)}(A,b)$ is not of full row rank and/or $\mathcal{O}_{(1:|Q|)}(c,A)$ is not of full column rank. It follows that rank $\mathcal{O}_{(1:|Q|)}(c,A)\mathcal{C}_{(1:|Q|)}(A,b) < |\mathbb{Q}|$. Therefore, from Theorem 4.2, there exists a realization of order smaller than $|\mathbb{Q}|$. Hence $(\mathbb{Q},\mathbb{Y},A,c,b)$ is not minimal. ■

3.2.4.2 Conversion from positive Mealy to minimal quasi Mealy model

In this section we describe a method to reduce a nonminimal quasi Mealy model. Because a positive Mealy model is typically nonminimal as a quasi Mealy model, the method can also be used to find a minimal quasi Mealy model which is equivalent to a given (minimal or nonminimal) positive Mealy model.

Given a nonminimal quasi Mealy model $(\mathbb{Q},\mathbb{Y},A,c,b)$, finding an equivalent minimal Mealy model can be done in two steps. The first step is to find an equivalent quasi Mealy model $(\mathbb{Q}^{\mathrm{c}},\mathbb{Y},A^{\mathrm{c}},c^{\mathrm{c}},b^{\mathrm{c}})$ for which $\mathcal{C}_{(1:|\mathbb{Q}|)}(A^{\mathrm{c}},b^{\mathrm{c}})$ has full row rank and in the second step one determines an equivalent quasi model $(\mathbb{Q}^{\mathrm{co}},\mathbb{Y},A^{\mathrm{co}},c^{\mathrm{co}},b^{\mathrm{co}})$ for which $\mathcal{C}_{(1:|\mathbb{Q}|)}(A^{\mathrm{co}},b^{\mathrm{co}})$ and $\mathcal{O}_{(1:|\mathbb{Q}|)}(c^{\mathrm{co}},A^{\mathrm{co}})$ have full row and full column rank respectively. We now describe both steps.

It is clear that for a quasi Mealy model $(\mathbb{Q},\mathbb{Y},A,c,b)$ every nonsingular matrix $T\in\mathbb{R}^{|\mathbb{Q}|\times|\mathbb{Q}|}$ gives rise to an equivalent model $(\mathbb{Q},\mathbb{Y},TAT^{-1},cT^{-1},Tb)$. Now let $(\mathbb{Q},\mathbb{Y},A,c,b)$ be such that

$$\text{rank } \mathcal{C}_{(1:|\mathbb{Q}|)}(A,b) = r < |\mathbb{Q}|.$$

Now there exists a nonsingular matrix R such that the equivalent Mealy model $(\mathbb{Q},\mathbb{Y},A'=RAR^{-1},c'=cR^{-1},b'=Rb)$ has the form

$$A'(\mathrm{y}) = \begin{array}{cc} \begin{array}{cc} {\scriptstyle |\mathbb{Q}|-r} & {\scriptstyle r} \end{array} & \\ \left[\begin{array}{c|c} * & 0 \\ \hline * & A^{\mathrm{c}}(\mathrm{y}) \end{array}\right] & \begin{array}{c} {\scriptstyle |\mathbb{Q}|-r} \\ {\scriptstyle r} \end{array} \end{array} \quad , \quad \forall \mathrm{y}\in\mathbb{Y},$$

$$b' = \begin{array}{cc} \left[\begin{array}{c} 0 \\ \hline b^{\mathrm{c}} \end{array}\right] & \begin{array}{c} {\scriptstyle |\mathbb{Q}|-r} \\ {\scriptstyle r} \end{array} \end{array}, \qquad c' = \begin{array}{c} \begin{array}{cc} {\scriptstyle |\mathbb{Q}|-r} & {\scriptstyle r} \end{array} \\ \left[\begin{array}{c|c} * & c^{\mathrm{c}} \end{array}\right] \end{array}.$$

Any realization of this form has the property that the r-th order subsystem $(\mathbb{Q}^{\mathrm{c}},\mathbb{Y},A^{\mathrm{c}},c^{\mathrm{c}},b^{\mathrm{c}})$ is equivalent to the system $(\mathbb{Q},\mathbb{Y},A,c,b)$ and that $\mathcal{C}_{(1:r)}(A^{\mathrm{c}},b^{\mathrm{c}})$ has full row rank.

Notice that

$$R\mathcal{C}_{(1:|\mathbb{Q}|)}(A,b) = \mathcal{C}_{(1:|\mathbb{Q}|)}(A',b') = \begin{array}{cc} \left[\begin{array}{cc} 0 & 0 \\ \mathcal{C}_{(1:r)}(A^{\mathrm{c}},b^{\mathrm{c}}) & * \end{array}\right] & \begin{array}{c} {\scriptstyle |\mathbb{Q}|-r} \\ {\scriptstyle r} \end{array} \end{array},$$

which suggests a procedure to compute the transformation R. Indeed, R is such that $R\mathcal{C}_{(1:|\mathbb{Q}|)}(A,b)$ has its first $|\mathbb{Q}|-r$ rows equal to 0. Such a transformation R can be found using the SVD.

We now describe an algorithm, inspired by [89], to find the transformation R directly from the system matrices without computing the $\mathcal{C}$-matrix. Therefore, we first define the matrix $P(A,b)$ for the model $(\mathbb{Q},\mathbb{Y},A,c,b)$ as

$$P(A,b) := \left[\begin{array}{ccccccc} A_{:,1}(\mathrm{y}^{(1)}) & \ldots & A_{:,1}(\mathrm{y}^{(|\mathbb{Y}|)}) & \ldots A_{:,|\mathbb{Q}|}(\mathrm{y}^{(1)}) & \ldots & A_{:,|\mathbb{Q}|}(\mathrm{y}^{(|\mathbb{Y}|)}) & b \end{array}\right].$$

Algorithm 3.1. *Given the quasi model $(\mathbb{Q},\mathbb{Y},A,c,b)$ with the corresponding matrix $P(A,b)$. Run the following steps.*

1. *Set $(\mathbb{Q}',\mathbb{Y},A',c',b') = (\mathbb{Q},\mathbb{Y},A,c,b)$, $P' = P$, $i = |\mathbb{X}|\cdot|\mathbb{Y}|+1$ and $j = |\mathbb{Q}|$.*
2. *If every element of $P'_{1:j,i}$ is equal to 0 then goto step 5.*
3. *Find a transformation $R_i = \begin{bmatrix} R_i' & 0 \\ & I_{|\mathbb{Q}|-j} \end{bmatrix}$ such that the vector $R_iP_{:,i}$ is of the form $\begin{bmatrix} 0 & \ldots & 0 & * & \ldots & * \end{bmatrix}^\top$ where the number of 0's is equal to $j-1$ and the number of scalars $*$ is equal to $|\mathbb{Q}|-j+1$. Transform $(\mathbb{Q}',\mathbb{Y},A',c',b')$ into $(\mathbb{Q}',\mathbb{Y},R_iA'R_i^{-1},c'R_i^{-1},R_ib')$ and recalculate the matrix P'.*
4. *Decrease i by 1, and decrease j by 1. If $j=0$ goto step 6. If $j>0$, go to step 2.*
5. *Increase i by 1. Goto step 2.*
6. *Calculate R as $R = R_iR_{i-1}\ldots R_2R_1$.*

So far we described a method to find, for a given nonminimal Mealy model $(\mathbb{Q},\mathbb{Y},A,c,b)$, an equivalent quasi Mealy model $(\mathbb{Q}^{\mathrm{c}},\mathbb{Y},A^{\mathrm{c}},c^{\mathrm{c}},b^{\mathrm{c}})$ for which $\mathcal{C}_{(1:|\mathbb{Q}^{\mathrm{c}}|)}(c^{\mathrm{c}},A^{\mathrm{c}})$ has full row rank. We now give a procedure to determine an equivalent quasi model $(\mathbb{Q}^{\mathrm{co}},\mathbb{Y},A^{\mathrm{co}},c^{\mathrm{co}},b^{\mathrm{co}})$ for which $\mathcal{C}_{(1:|\mathbb{Q}^{\mathrm{co}}|)}(A^{\mathrm{co}},b^{\mathrm{co}})$ and $\mathcal{O}_{(1:|\mathbb{Q}^{\mathrm{co}}|)}(c^{\mathrm{co}},A^{\mathrm{co}})$ have full row and full column rank respectively. For this second step, suppose that

$$\operatorname{rank}\ \mathcal{O}_{(1:|\mathbb{Q}^{\mathrm{c}}|)}(c^{\mathrm{c}},A^{\mathrm{c}}) = s < |\mathbb{Q}^{\mathrm{c}}| = r,$$

for $(\mathbb{Q}^{\mathrm{c}},\mathbb{Y},A^{\mathrm{c}},c^{\mathrm{c}},b^{\mathrm{c}})$. Then it can be shown that there exists a nonsingular matrix S such that the equivalent Mealy model $(\mathbb{Q}^{\mathrm{c}},\mathbb{Y},A^{\mathrm{c}\prime} = SA^{\mathrm{c}}S^{-1}, c^{\mathrm{c}\prime} = c^{\mathrm{c}}S^{-1}, b^{\mathrm{c}\prime} = Sb)$ has the form

$$A^{\mathrm{c}\prime}(\mathrm{y}) = \begin{array}{c} \begin{array}{cc} \scriptstyle r-s & \scriptstyle s \end{array} \\ \left[\begin{array}{c|c} * & * \\ \hline 0 & A^{\mathrm{co}}(\mathrm{y}) \end{array}\right] \end{array} \begin{array}{l} \scriptstyle r-s \\ \scriptstyle s \end{array}, \quad \forall \mathrm{y}\in\mathbb{Y},$$

$$b^{\mathrm{c}\prime} = \left[\begin{array}{c} * \\ \hline b^{\mathrm{c}} \end{array}\right] \begin{array}{l} \scriptstyle r-s \\ \scriptstyle s \end{array}, \qquad c^{\mathrm{c}\prime} = \begin{array}{c} \begin{array}{cc} \scriptstyle r-s & \scriptstyle s \end{array} \\ \left[\begin{array}{c|c} 0 & c^{\mathrm{c}} \end{array}\right] \end{array}.$$

It holds for this realization that the s-th order subsystem $(\mathbb{Q}^{\mathrm{co}}, \mathbb{Y}, A^{\mathrm{co}}, c^{\mathrm{co}}, b^{\mathrm{co}})$ is equivalent with the system $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ and that $\mathcal{C}(A^{\mathrm{co}}, b^{\mathrm{co}})$ has full row rank and that $\mathcal{O}(c^{\mathrm{co}}, A^{\mathrm{co}})$ has full column rank, i.e. the subsystem $(\mathbb{Q}^{\mathrm{co}}, \mathbb{Y}, A^{\mathrm{co}}, c^{\mathrm{co}}, b^{\mathrm{co}})$ is minimal. The procedure to find the transformation S is dual to the procedure to find the transformation R as decribed before.

By combining both steps, we have that for every nonminimal Mealy model $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ there exists a transformation T such that

$$
\begin{aligned}
TA(\mathsf{y})T^{-1} &= \begin{array}{cc} \begin{array}{ccc} {\scriptstyle |\mathbb{Q}|-r} & {\scriptstyle r-s} & {\scriptstyle s} \end{array} & \\ \left[\begin{array}{c|c|c} * & 0 & 0 \\ \hline * & * & * \\ \hline * & 0 & A^{\mathrm{co}}(\mathsf{y}) \end{array}\right] & \begin{array}{c} {\scriptstyle |\mathbb{Q}|-r} \\ {\scriptstyle r-s} \\ {\scriptstyle s} \end{array} \end{array}, \qquad \forall \mathsf{y} \in \mathbb{Y}, \\
Tb &= \left[\begin{array}{c} 0 \\ \hline * \\ \hline b^{\mathrm{co}} \end{array}\right] \begin{array}{c} {\scriptstyle |\mathbb{Q}|-r} \\ {\scriptstyle r-s} \\ {\scriptstyle s} \end{array}, \qquad cT^{-1} = \begin{array}{c} \begin{array}{ccc} {\scriptstyle |\mathbb{Q}|-r} & {\scriptstyle r-s} & {\scriptstyle s} \end{array} \\ \left[\begin{array}{c|c|c} * & 0 & c^{\mathrm{co}} \end{array}\right] \end{array}.
\end{aligned}
\tag{3.8}
$$

The transformation T can be computed from R and S by

$$
T = \begin{bmatrix} I & 0 \\ 0 & S \end{bmatrix} R.
$$

Example 3.4. *Given the Mealy model $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ of Example 3.2, Algorithm 3.1 yields a transformation T such that*

$$
\begin{aligned}
T\Pi(a)T^{-1} &= \left[\begin{array}{r|rrr} 0.000 & 0 & 0 & 0 \\ \hline 0.000 & 0.000 & 0.000 & 0.000 \\ 0.040 & -0.050 & 0.050 & -0.040 \\ 0.050 & 0.040 & -0.040 & 0.050 \end{array}\right], \\
T\Pi(b)T^{-1} &= \left[\begin{array}{r|rrr} 0.000 & 0 & 0 & 0 \\ \hline 0.405 & 0.405 & 0.405 & 0.405 \\ 0.005 & -0.005 & 0.005 & -0.005 \\ 0.400 & 0.410 & 0.400 & 0.410 \end{array}\right], \\
T\Pi(c)T^{-1} &= \left[\begin{array}{r|rrr} 0.000 & 0 & 0 & 0 \\ \hline -0.360 & 0.450 & 0.450 & -0.360 \\ 0.000 & 0.000 & 0.000 & 0.000 \\ 0.450 & -0.360 & -0.360 & 0.450 \end{array}\right], \\
T\Pi(d)T^{-1} &= \left[\begin{array}{r|rrr} 0.000 & 0 & 0 & 0 \\ \hline -0.045 & 0.045 & 0.045 & -0.045 \\ -0.045 & -0.045 & 0.045 & 0.045 \\ 0.000 & -0.090 & 0.000 & 0.060 \end{array}\right], \\
\pi(1)T^{-1} &= \left[\begin{array}{r|rrr} -0.400 & 0.000 & 0.000 & -0.500 \end{array}\right], \\
Te &= \left[\begin{array}{r|rrr} 0 & 0.000 & 0.000 & 2.000 \end{array}\right]^{\top}.
\end{aligned}
$$

Hence a minimal quasi Mealy model equivalent to the given positive Mealy model is given by $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ where $\mathbb{X} = \{1, 2, 3\}$ and

$$
A(a) = \begin{bmatrix} 0.000 & 0.000 & 0.000 \\ -0.050 & 0.050 & -0.040 \\ 0.040 & -0.040 & 0.050 \end{bmatrix}, \quad A(b) = \begin{bmatrix} 0.405 & 0.405 & 0.405 \\ -0.005 & 0.005 & -0.005 \\ 0.410 & 0.400 & 0.410 \end{bmatrix},
$$

$$
A(c) = \begin{bmatrix} 0.450 & 0.450 & -0.360 \\ 0.000 & 0.000 & 0.000 \\ -0.360 & -0.360 & 0.450 \end{bmatrix}, \quad A(d) = \begin{bmatrix} 0.045 & 0.045 & -0.045 \\ -0.045 & 0.045 & 0.045 \\ -0.090 & 0.000 & 0.060 \end{bmatrix},
$$

$$
\begin{aligned}
c &= \begin{bmatrix} 0.000 & 0.000 & -0.500 \end{bmatrix}, \\
b &= \begin{bmatrix} 0.000 & 0.000 & 2.000 \end{bmatrix}^{\top}.
\end{aligned}
$$

We have shown that the model of Example 3.3 is a minimal quasi HMM equivalent to the positive HMM of Example 3.2. □

3.3 Equivalence of HMMs

In this section we consider the equivalence problem for hidden Markov models: given a minimal HMM, under which conditions is another Mealy model equivalent to it and how is the set of all equivalent HMMs characterized. The situation is analogous to the situation for linear stochastic models (see Figure 3.2). Given a hidden Markov model (Mealy, quasi Mealy or Moore), an equivalent model can be obtained by permuting the states. This is the analogue of the equivalence transformation for linear stochastic models. However, as decribed in this section, much more equivalents are possible. These equivalents are the analogue of the fact that a convex set of state covariance matrices gives rise to equivalent linear stochastic models.

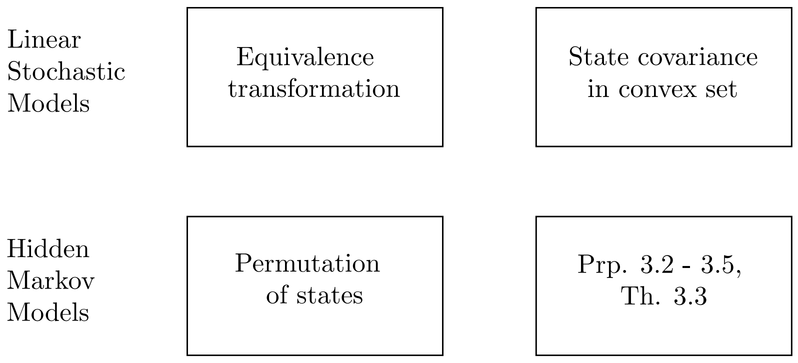


Figure 3.2: *The equivalence problem for the different types of hidden Markov models (Mealy, quasi Mealy and Moore hidden Markov models) is analogous to the equivalence problem for linear stochastic models. Given a hidden Markov model, an equivalent model can be obtained by permuting the states. This is the analogue of the equivalence transformation for linear stochastic models. However, much more equivalents are possible: see Proposition 3.2, Proposition 3.3, Proposition 3.4, Theorem 3.3 and Proposition 3.5. These equivalents are the analogue of the fact that a convex set of state covariance matrices gives rise to equivalent linear stochastic models.*

In Section 3.3.1 we consider the equivalence problem for Mealy HMMs and in Section 3.3.2 for Moore HMMs. In Section 3.3.3 we give a summary of the equivalence problem for hidden Markov models.

3.3.1 Equivalence of Mealy HMMs

In this section we investigate the equivalence problem both for the quasi Mealy case (Section 3.3.1.1) as for the positive Mealy case (Section 3.3.1.2).

3.3.1.1 Equivalence of quasi Mealy HMMs

Given a certain quasi Mealy model, one can always obtain an equivalent model by permuting the states. However, there are many more equivalent models than only the ones obtained by permuting states (Figure 3.2). For quasi Mealy HMMs the equivalence of realizations is described by the following proposition [112,113].

Proposition 3.2. *Consider a minimal quasi Mealy model* $(\mathbb{Q}, \mathbb{Y}, A, c, b)$. *The quasi Mealy model* ($\mathbb{Q}$, $\mathbb{Y}$, A', c', b') *is an equivalent model if and only if there exists a nonsingular matrix* T*, such that*

$$\begin{array}{l} \forall \boldsymbol{y} \in \mathbb{Y} : A'(\boldsymbol{y}) = TA(\boldsymbol{y})T^{-1}, \\ c' = cT^{-1}, \\ b' = Tb. \end{array} \tag{3.9}$$

It can be proven that the set of all quasi Mealy models equivalent to a given quasi Mealy model forms a *semi-algebraic set* (see Appendix B). This set can be *constructed* (see Appendix B), which means that the set can be described as a finite union of subsets.

3.3.1.2 Equivalence of positive Mealy HMMs

Again, given a certain positive Mealy model, one can always obtain an equivalent model by permuting the states. However, there are many more equivalent models than only the ones obtained by permuting states (Figure 3.2). In this section we describe the complete equivalence sets for positive Mealy models. We first deal with a special situation where the Mealy model is minimal as a quasi Mealy model. Next, we consider the most general case, where the order of the minimal Mealy model is larger than or equal to the order of a minimal equivalent quasi Mealy model.

For the situation where the Mealy model is minimal as a quasi Mealy model, we prove the following proposition.

Proposition 3.3. *Given a minimal Mealy model* $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ *which is also minimal as a quasi Mealy model. Then the model* $(\mathbb{X}, \mathbb{Y}, \Pi', \pi'(1))$ *is an equivalent Mealy model if and only if there exists a nonsingular matrix* $T \in \mathbb{R}^{|\mathbb{X}| \times |\mathbb{X}|}$ *such that*

$$\begin{array}{l} \forall \boldsymbol{y} \in \mathbb{Y} : \Pi'(\boldsymbol{y}) = T\Pi(\boldsymbol{y})T^{-1}, \\ \pi'(1) = \pi(1)T^{-1}, \\ Te = e, \\ \forall \boldsymbol{y} \in \mathbb{Y} : \Pi'(\boldsymbol{y}) \geq 0, \\ \pi'(1) \geq 0. \end{array} \tag{3.10}$$

Proof: The proposition basically follows from Proposition 3.2. For the $\Leftarrow$ part, it remains to be proven that $(\mathbb{X}, \mathbb{Y}, \Pi', \pi'(1))$ fullfilling (3.10) satisfies the consistency conditions of Mealy models. To see this, first note that $\pi'(1)e =$

$\pi(1)T^{-1}e = \pi(1)e = 1$. Next, from

$$\sum_{\mathrm{y}\in\mathbb{Y}} \Pi'(\mathrm{y})e = \sum_{\mathrm{y}\in\mathbb{Y}} T\Pi(\mathrm{y})T^{-1}e = T\left(\sum_{\mathrm{y}\in\mathbb{Y}} \Pi(\mathrm{y})\right)e = Te = e,$$

it follows that $\sum_{\mathrm{y}\in\mathbb{Y}} \Pi'(\mathrm{y})$ is a stochastic matrix. ■

In the same way as in the proof of the above proposition, it can be proven that if $(\mathbb{X},\mathbb{Y},\Pi,\pi(1))$ is stationary (i.e. $\pi(1)\sum_{\mathrm{y}\in\mathbb{Y}}\Pi(\mathrm{y}) = \pi(1)$), then $(\mathbb{X},\mathbb{Y},\Pi',\pi'(1))$ is also stationary.

It can be proven that the set of all positive Mealy models equivalent to a given positive Mealy model, that is minimal as a quasi model, forms a *semi-algebraic set* (see Appendix B). This set can be *constructed* (see Appendix B), which means that the set can be described as a finite union of subsets.

For the most general situation, when the order of the minimal Mealy model is larger than or equal to the minimal quasi Mealy order, we prove the following proposition.

Proposition 3.4. *Given a minimal Mealy model* $(\mathbb{X},\mathbb{Y},\Pi,\pi(1))$. *Then the model* $(\mathbb{X},\mathbb{Y},\Pi',\pi'(1))$ *is an equivalent Mealy model if and only if there exist positive scalars* r, r' *and* s, *nonsingular matrices* T *and* $T' \in \mathbb{R}^{|\mathbb{X}|\times|\mathbb{X}|}$, *and a minimal quasi Mealy model* $(\{1,\ldots,s\},\mathbb{Y},A^{co},c^{co},b^{co})$ *such that*

$$\begin{array}{l}
\forall y\in\mathbb{Y}: T\Pi(y)T^{-1} = \begin{array}{cc} \begin{array}{ccc} {\scriptstyle |\mathbb{X}|-r} & {\scriptstyle r-s} & {\scriptstyle s} \end{array} & \\ \left[\begin{array}{c|c|c} * & 0 & 0 \\ \hline * & * & * \\ \hline * & 0 & A^{co}(y) \end{array}\right] & \begin{array}{c} {\scriptstyle |\mathbb{X}|-r} \\ {\scriptstyle r-s} \\ {\scriptstyle s} \end{array} \end{array}, \\[2ex]
\forall y\in\mathbb{Y}: T'\Pi'(y)T'^{-1} = \begin{array}{cc} \begin{array}{ccc} {\scriptstyle |\mathbb{X}|-r'} & {\scriptstyle r'-s} & {\scriptstyle s} \end{array} & \\ \left[\begin{array}{c|c|c} A_{(1,1)}(y) & 0 & 0 \\ \hline A_{(2,1)}(y) & A_{(2,2)}(y) & A_{(2,3)}(y) \\ \hline A_{(3,1)}(y) & 0 & A^{co}(y) \end{array}\right] & \begin{array}{c} {\scriptstyle |\mathbb{X}|-r'} \\ {\scriptstyle r'-s} \\ {\scriptstyle s} \end{array} \end{array}, \\[2ex]
\pi(1)T^{-1} = \begin{array}{c} \begin{array}{ccc} {\scriptstyle |\mathbb{X}|-r} & {\scriptstyle r-s} & {\scriptstyle s} \end{array} \\ \left[\begin{array}{c|c|c} * & 0 & c^{co} \end{array}\right] \end{array}, \qquad Te = \left[\begin{array}{c} 0 \\ \hline * \\ \hline b^{co} \end{array}\right] \begin{array}{c} {\scriptstyle |\mathbb{X}|-r} \\ {\scriptstyle r-s} \\ {\scriptstyle s} \end{array}, \\[2ex]
\pi'(1)T'^{-1} = \begin{array}{c} \begin{array}{ccc} {\scriptstyle |\mathbb{X}|-r'} & {\scriptstyle r'-s} & {\scriptstyle s} \end{array} \\ \left[\begin{array}{c|c|c} c_{(1)} & 0 & c^{co} \end{array}\right] \end{array}, \qquad T'e = \left[\begin{array}{c} 0 \\ \hline b_{(2)} \\ \hline b^{co} \end{array}\right] \begin{array}{c} {\scriptstyle |\mathbb{Q}|-r'} \\ {\scriptstyle r'-s} \\ {\scriptstyle s} \end{array}, \\[2ex]
\forall y\in\mathbb{Y}: \Pi'(y) \geq 0, \\
\pi'(1) \geq 0, \\
\sum_y \left[\begin{array}{cc} A_{(2,2)}(y) & A_{(2,3)}(y) \end{array}\right] \left[\begin{array}{c} b_{(2)} \\ b^{co} \end{array}\right] = b_{(2)}.
\end{array} \tag{3.11}$$

Proof: The proof follows from the procedure for obtaining a minimal quasi Mealy model from a positive Mealy model described in Section 3.2.4.2. For the

$\Leftarrow$ part, it remains to be proven that $(\mathbb{X}, \mathbb{Y}, \Pi', \pi'(1))$ fullfilling (3.11) satisfies the consistency conditions of Mealy models. To prove that $\pi'(1)$ has element sum equal to 1, one can see that $\pi'(1)e = c^{\text{co}}b^{\text{co}} = \pi(1)e = 1$. Next, $\sum_{\mathsf{y}} \Pi'(\mathsf{y})e = e$ if and only if

$$\sum_{\mathsf{y}} \begin{bmatrix} A_{(1,1)}(\mathsf{y}) & 0 & 0 \\ A_{(2,1)}(\mathsf{y}) & A_{(2,2)}(\mathsf{y}) & A_{(2,3)}(\mathsf{y}) \\ A_{(3,1)}(\mathsf{y}) & 0 & A^{\text{co}}(\mathsf{y}) \end{bmatrix} \begin{bmatrix} 0 \\ b_{(2)} \\ b^{\text{co}} \end{bmatrix} = \begin{bmatrix} 0 \\ b_{(2)} \\ b^{\text{co}} \end{bmatrix}. \tag{3.12}$$

This condition is true if $\sum_{\mathsf{y}} A^{\text{co}}(\mathsf{y})b^{\text{co}} = b^{\text{co}}$, which is true because $\sum_{\mathsf{y}} \Pi(\mathsf{y})e = e$. ■

If the original model is stationary (i.e. $\pi(1)\sum_{\mathsf{y}\in\mathbb{Y}} \Pi(\mathsf{y}) = \pi(1)$), then an equivalent model is not necessary stationary. By adding the condition

$$\begin{bmatrix} c_{(1)} & c^{\text{co}} \end{bmatrix} \sum_{\mathsf{y}} \begin{bmatrix} A_{(1,1)}(\mathsf{y}) \\ A_{(3,1)}(\mathsf{y}) \end{bmatrix} = c_{(1)}. \tag{3.13}$$

to (3.11), a stationary model will give rise to an equivalent stationary model.

To check whether two (positive or quasi) Mealy models are equivalent, one can use Proposition 3.4 as follows. First find for both Mealy models an equivalent minimal quasi Mealy model using the procedure of Section 3.2.4.2. Next check whether there exists a transformation that transforms the first quasi model into the second quasi model. If such a transformation exists, the original models are equivalent, otherwise, they are not equivalent.

Example 3.5. *Suppose we are given the positive Mealy model* $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ *of Example 3.3, and the quasi Mealy model* $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ *defined below. The question is to check whether both models are equivalent or not.*

$$A(a) = \begin{bmatrix} 0.050 & -0.018 & 0.005 \\ 0.082 & -0.047 & 0.012 \\ 0.662 & -0.378 & 0.097 \end{bmatrix}, \qquad A(b) = \begin{bmatrix} 0.410 & -0.189 & -0.001 \\ -0.870 & 0.400 & 0.004 \\ 0.092 & -0.042 & 0.010 \end{bmatrix},$$

$$A(c) = \begin{bmatrix} 0.450 & 0.166 & 0.001 \\ 0.783 & 0.451 & 0.003 \\ -0.021 & -0.011 & -0.000 \end{bmatrix}, \qquad A(d) = \begin{bmatrix} 0.090 & 0.041 & -0.005 \\ 0.006 & 0.002 & 0.011 \\ -0.747 & -0.343 & 0.087 \end{bmatrix},$$

$$\begin{aligned} c &= \begin{bmatrix} -1.1759 & 0.001 & 0.000 \end{bmatrix}, \\ b &= \begin{bmatrix} -0.850 & 0.001 & 0.014 \end{bmatrix}^\top . \end{aligned}$$

Using the procedure of Section 3.2.4.2, one can compute a minimal quasi HMM $(\mathbb{Q}, \mathbb{Y}, A^{(\text{co})}, c^{(\text{co})}, b^{(\text{co})})$ *equivalent to the positive model* $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ *(see Example 3.4). Subsequently, it is easy to verify that there exists a transformation* T *between* $(\mathbb{Q}, \mathbb{Y}, A^{(\text{co})}, c^{(\text{co})}, b^{(\text{co})})$ *and* $(\mathbb{Q}, \mathbb{Y}, A, c, b)$*. We conclude that the given positive and quasi HMMs are equivalent.* □

Again, it can be proven that the set of all positive Mealy models equivalent to a given positive Mealy model forms a *semi-algebraic set* (see Appendix B) which can be *constructed* (see Appendix B).

3.3.2 Equivalence of Moore HMMs

In this section we investigate the set of equivalent Moore HMMs. Clearly, we obtain a model equivalent to a given positive Moore model by permuting the states of the original model. However, other equivalent models are possible (Figure 3.2). In this section we describe the complete set of equivalent models to a Moore model. We first deal with the case where the Moore model is minimal as a quasi Mealy model. Next we consider the general case, where the order of the minimal Moore model is larger than or equal to the order of a minimal equivalent quasi Mealy model.

Consider a Moore model that is minimal as a quasi Mealy model. We show that under certain conditions every equivalent Moore model corresponds to a permutation of the states of the given model.

Theorem 3.2. *Let* $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, B, \pi(1))$ *be a Moore HMM, which is minimal as a quasi Mealy model. Suppose the state transition matrix* $\Pi_{\mathbb{X}}$ *has full rank and all states of the Moore model have a different output distribution (i.e. no two rows of* B *are equal to each other). Then every minimal Moore model that is equivalent to the given Moore model is obtained by permuting the states of the original model.*

Proof: Suppose that $(\mathbb{X}, \mathbb{Y}, \Pi'_{\mathbb{X}}, B', \pi'(1))$ is equivalent to and of the same order as $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, B, \pi(1))$. Then from Proposition 3.2, there exists a nonsingular matrix T such that

$$\begin{aligned} &\forall \mathrm{y} \in \mathbb{Y} : \operatorname{diag}(\beta'(\mathrm{y}))\Pi'_{\mathbb{X}} = T \operatorname{diag}(\beta(\mathrm{y}))\Pi_{\mathbb{X}} T^{-1}, \\ &\pi'(1) = \pi(1) T^{-1}, \\ &e = Te. \end{aligned} \tag{3.14}$$

Since $\Pi_{\mathbb{X}}$ has full rank, it follows that $\Pi'_{\mathbb{X}}$ has full rank. So it follows from (3.14) that there exist nonsingular matrices T and S such that

$$\begin{aligned} &\forall \mathrm{y} \in \mathbb{Y} : \operatorname{diag}(\beta'(\mathrm{y})) = T \operatorname{diag}(\beta(\mathrm{y})) S^{-1}, \\ &\Pi'_{\mathbb{X}} = S \Pi_{\mathbb{X}} T^{-1}. \end{aligned}$$

For the model $(\mathbb{X}, \mathbb{Y}, \Pi'_{\mathbb{X}}, B', \pi'(1))$, it must hold that $\sum_{\mathrm{y} \in \mathbb{Y}} T \operatorname{diag}(\beta(\mathrm{y})) S^{-1} = I$, which gives $T = S$. It follows that

$$\forall \mathrm{y} \in \mathbb{Y} : \operatorname{diag}(\beta'(\mathrm{y})) = T \operatorname{diag}(\beta(\mathrm{y})) T^{-1}. \tag{3.15}$$

Together with $Te = e$ and with the fact that all states of the Moore model have a different output distribution, this allows to conclude that T can only be equal to a permutation matrix. ■

If there exist states with the same output distribution, and if all the other conditions of Theorem 3.2 are fulfilled, then there exists a set of equivalent Moore models (apart from the models obtained by permuting the states).

Theorem 3.3. *Let* $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, B, \pi(1))$ *be a Moore HMM, which is minimal as a quasi Mealy model. Suppose the state transition matrix* $\Pi_{\mathbb{X}}$ *has full rank and* *the first r states have the same output distribution (i.e. the first r rows of B are equal to each other). Then the model $(\mathbb{X}', \mathbb{Y}, \Pi'_{\mathbb{X}}, B', \pi'(1))$ is an equivalent Moore model if and only if there exists a permutation matrix $P \in \mathbb{R}^{|\mathbb{Q}|\times|\mathbb{Q}|}$ and a nonsingular matrix $T \in \mathbb{R}^{r\times r}$ such that*

$$\begin{aligned}
&\Pi'_{\mathbb{X}} = P \begin{bmatrix} T & 0 \\ 0 & I \end{bmatrix} \Pi_{\mathbb{X}} \begin{bmatrix} T^{-1} & 0 \\ 0 & I \end{bmatrix} P^{\top}, \\
&B' = PB, \\
&\pi'(1) = \pi(1) \begin{bmatrix} T^{-1} & 0 \\ 0 & I \end{bmatrix} P^{\top}, \\
&Te = e, \\
&\Pi'_{\mathbb{X}}, B', \pi'(1) \geq 0.
\end{aligned}$$

Proof: The theorem follows from Equation (3.15) and from the fact that first r states have the same output distribution. ■

Of course, an extended version of Theorem 3.3 holds, in case there are different sets of states with the same output distribution.

Example 3.6. *Consider the Moore model $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, B, \pi(1))$ with*

$$\Pi_{\mathbb{X}} = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.3 & 0.3 & 0.4 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}, \qquad B = \begin{bmatrix} 0.3 & 0.3 & 0.4 \\ 0.1 & 0.1 & 0.8 \\ 0.2 & 0.6 & 0.2 \end{bmatrix},$$
$$\pi(1) = \begin{bmatrix} 0.5405 & 0.1622 & 0.2973 \end{bmatrix}.$$

It can be shown[4] that the order of a minimal equivalent quasi Mealy model equals 3. Hence the Moore model is minimal as a quasi Mealy model. In addition all rows of B are different and $\Pi_{\mathbb{X}}$ has full rank. We conclude from Theorem 3.2 that the only way to obtain a minimal Moore equivalent to the given model is by permuting the states. □

We now consider the general case where the order of the Moore model is larger than or equal to the order of a minimal equivalent quasi Mealy model.

Proposition 3.5. *Consider a minimal Moore model $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, B, \pi(1))$. The Moore model $(\mathbb{X}, \mathbb{Y}, \Pi'_{\mathbb{X}}, B', \pi'(1))$ is an equivalent model if and only if there exist positive scalars r, r' and s, nonsingular matrices T and $T' \in \mathbb{R}^{|\mathbb{X}|\times|\mathbb{X}|}$, and a*

[4]By computing a Mealy model $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ equivalent to the Moore model and by subsequently applying the test for minimality of Mealy models (explained in Section 3.2.4.1).

minimal quasi Mealy model $(\{1,\dots,s\},\mathbb{Y},A^{co},c^{co},b^{co})$ *such that*

$$\forall y\in\mathbb{Y}: T\,\mathrm{diag}(\beta(y))\Pi_{\mathbb{X}}T^{-1}=\begin{array}{c c}\begin{array}{ccc} {\scriptstyle |\mathbb{X}|-r} & {\scriptstyle r-s} & {\scriptstyle s}\end{array} & \\ \left[\begin{array}{c|c|c} * & 0 & 0\\ \hline * & * & *\\ \hline * & 0 & A^{co}(y)\end{array}\right] & \begin{array}{c}{\scriptstyle |\mathbb{X}|-r}\\ {\scriptstyle r-s}\\ {\scriptstyle s}\end{array}\end{array},$$

$$\forall y\in\mathbb{Y}: T'\,\mathrm{diag}(\beta'(y))\Pi'_{\mathbb{X}}T'^{-1}=\begin{array}{c c}\begin{array}{ccc} {\scriptstyle |\mathbb{X}|-r'} & {\scriptstyle r'-s} & {\scriptstyle s}\end{array} & \\ \left[\begin{array}{c|c|c} A_{(1,1)}(y) & 0 & 0\\ \hline A_{(2,1)}(y) & A_{(2,2)}(y) & A_{(2,3)}(y)\\ \hline A_{(3,1)}(y) & 0 & A^{co}(y)\end{array}\right] & \begin{array}{c}{\scriptstyle |\mathbb{X}|-r'}\\ {\scriptstyle r'-s}\\ {\scriptstyle s}\end{array}\end{array},$$

$$\pi(1)T^{-1}=\begin{array}{c}\begin{array}{ccc}{\scriptstyle |\mathbb{X}|-r} & {\scriptstyle r-s} & {\scriptstyle s}\end{array}\\ \left[\begin{array}{c|c|c} * & 0 & c^{co}\end{array}\right]\end{array},\qquad Te=\begin{array}{cc}\left[\begin{array}{c}0\\ \hline *\\ \hline b^{co}\end{array}\right] & \begin{array}{c}{\scriptstyle |\mathbb{X}|-r}\\ {\scriptstyle r-s}\\ {\scriptstyle s}\end{array}\end{array},$$

$$\pi'(1)T'^{-1}=\begin{array}{c}\begin{array}{ccc}{\scriptstyle |\mathbb{X}|-r'} & {\scriptstyle r'-s} & {\scriptstyle s}\end{array}\\ \left[\begin{array}{c|c|c} c_{(1)} & 0 & c^{co}\end{array}\right]\end{array},\qquad T'e=\begin{array}{cc}\left[\begin{array}{c}0\\ \hline b_{(2)}\\ \hline b^{co}\end{array}\right] & \begin{array}{c}{\scriptstyle |\mathbb{X}|-r'}\\ {\scriptstyle r'-s}\\ {\scriptstyle s}\end{array}\end{array},$$

$$\begin{array}{l}\Pi'_{\mathbb{X}},\ B',\ \pi'(1)\geq 0,\\ B'e=e,\\ \sum_y\left[\begin{array}{cc}A_{(2,2)}(y) & A_{(2,3)}(y)\end{array}\right]\left[\begin{array}{c}b_{(2)}\\ b^{co}\end{array}\right]=b^{(2)}.\end{array}\tag{3.16}$$

Proof: The proof is analogous to the proof of Proposition 3.3. ■
The same remark concerning stationarity holds as for Proposition 3.4.

The set of all positive Moore models equivalent to a given positive Moore model forms a *semi-algebraic set* which can be *constructed* (see Appendix B).

3.3.3 Summary of equivalence of HMMs

We now summarize the results concerning the equivalence sets of quasi and positive Mealy HMMs and of Moore HMMs.

In Figure 3.3, we consider the relation between the sets of quasi and positive Mealy HMMs. From Proposition 3.2, it follows that there exists a set of equivalent quasi Mealy models. The order of an equivalent minimal positive Mealy model is either equal to the order of the minimal quasi Mealy model (Figure 3.3(a)), or larger than the order of the minimal quasi Mealy model (Figure 3.3(b)). In the first case the set of equivalent positive Mealy models is described by Proposition 3.3, while in the second case the set is described by Proposition 3.4.

In Figure 3.4, we consider the relation between the sets of quasi Mealy and positive Moore HMMs. First of all, as described by Proposition 3.2 there exists a set of equivalent quasi Mealy models. The order of a minimal equivalent Moore model is either equal to the order of the minimal quasi Mealy model (Figure 3.4(a)-(b)), or larger than the order of the minimal quasi Mealy model (Figure

3.4(c)). If in the first case, every state of the Moore model has a different output distribution and the transition matrix has full rank, then the Moore model is unique (up to a permutation of the states) by Theorem 3.2 (Figure 3.4(a)). Otherwise if some states have the same output distribution, there exists a set of equivalent Moore models described by Theorem 3.3 (Figure 3.4(b)). If the order of a minimal equivalent Moore model is larger than the order of the quasi Mealy model, then there exists a class of Moore models described by Proposition 3.5.

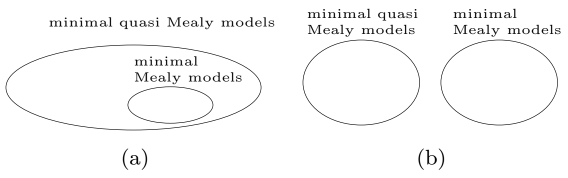


Figure 3.3: *This figure summarizes the equivalence classes for Mealy HMMs. There exists a set of equivalent quasi Mealy models. The order of an equivalent minimal positive Mealy model is either equal to the order of the minimal quasi Mealy model (Subfigure (a)), or larger than the order of the minimal quasi Mealy model (Subfigure (b)). In the first case the set of equivalent positive Mealy models is described by Proposition 3.3, while in the second case the set is described by Proposition 3.4.*

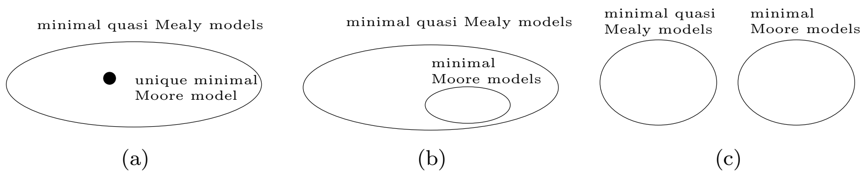

minimal quasi Mealy models
unique minimal Moore model
minimal quasi Mealy models
minimal Moore models
minimal quasi Mealy models
minimal Moore models
(a)
(b)
(c)

Figure 3.4: *This figure summarizes the equivalence classes of quasi Mealy and positive Moore models. There exists a set of equivalent quasi Mealy models. The order of a minimal equivalent Moore model is either equal to the order of the minimal quasi Mealy model (Subfigure (a)-(b)), or larger than the order of the minimal quasi Mealy model (Subfigure (c)). If in the first case, every state of the Moore model has a different output distribution and the transition matrix has full rank, then the Moore model is unique (up to a permutation of the states) by Theorem 3.2 (Subfigure (a)). Otherwise if some states have the same output distribution, there exists a set of equivalent Moore models described by Theorem 3.3 (Subfigure (b)). If the order of a minimal equivalent Moore model is larger than the order of the quasi Mealy model, then there exists a class of Moore models described by Proposition 3.5 (Subfigure (c)).*

3.4 Linear stochastic models

Hidden Markov models are closely related to linear stochastic models. We will return to the close relation between hidden Markov models and linear stochastic models in many of the following chapters. In this section we introduce linear stochastic models.

A *(time-homogeneous) linear stochastic model* is defined as

$$\begin{aligned} x(t+1) &= Ax(t) + w(t), \\ y(t) &= Cx(t) + v(t), \end{aligned}$$

where y is the *output process* taking values in the *output space* $\mathbb{R}^p$ and x is the *state process* taking values in the *state space* $\mathbb{R}^n$. The random variables $w(t)$ and $v(t)$ are zero mean, white Gaussian vector variables with covariance matrix

$$E(\begin{bmatrix} w(p) \\ v(p) \end{bmatrix} \begin{bmatrix} w(q)^\top & v(q)^\top \end{bmatrix}) = \begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} \delta(p,q),$$

where

$$\begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} \succeq 0,$$

$\delta(p,q)$ is the Kronecker delta and $x(0)$ is a zero mean random variable, independent of w and v, with covariance $E(x(0)x(0)^\top) = P$. It is clear that P, Q and R need to be positive definite. The dimension of the state space n is called the *order* of the linear stochastic model. Throughout the thesis, we only consider linear stochastic models for which the process x is stationary, i.e.

$$\begin{aligned} E(x(t)) &= 0, \\ E(x(t)x(t)^\top) &=: P, \end{aligned}$$

where the *state covariance matrix* P is independent of the time t. This implies that all eigenvalues of A are strictly inside the unit circle. It can be shown for stationary models that

$$P = APA^\top + Q.$$

From the fact that x is stationary, it follows that y is also stationary. A stationary linear stochastic model is denoted as (A, C, P, Q, R, S).

First notice that a linear stochastic model as defined above is the analogue of a Mealy HMM since the generation of the next state given the present state and the generation of the output given the present state are dependent events. A linear stochastic model with $S = 0$ is the analogue of a Moore HMM. Analogous to the quasi HMM case, one could call $(A, C, P^{(q)}, Q^{(q)}, R^{(q)}, S^{(q)})$, with $P^{(q)}$, $Q^{(q)}$ and $R^{(q)}$ not necessary positive definite, a quasi linear stochastic model. However, working with quasi linear stochastic models does not yield much advantage as we will show in Section 4.5 that the positive definiteness of P, Q and R comes for free in the linear stochastic realization algorithm. This is a difference between linear stochastic models and hidden Markov models.

Let $\Lambda : \mathbb{Z}_+ \mapsto \mathbb{R}^{p\times p}$ be the *autocovariances* of y, defined as $\Lambda(t) = E(y(\tau+t)y(\tau)^\top)$. The autocovariances *generated* by the linear stochastic model (A, C, P, Q, R, S) are given by

$$\begin{aligned}\Lambda(0) &= CPC^\top + R,\\ \Lambda(t) &= CA^{t-1}G,\end{aligned}$$

where G is defined as $G := E(x(t+1)y(t)^\top))$ and calculated as $G = APC^\top + S$.

We can now formulate the linear stochastic realization problem and the minimal linear stochastic realization problem. Notice that we only consider a Mealy version of the realization problem, in contrast to realization for HMMs where we made a distinction between the Moore and Mealy realization problem.

Problem 3.7 (linear stochastic realization problem)**.** *Given autocovariances $\Lambda : \mathbb{Z}_+ \mapsto \mathbb{R}^{p\times p}$. Find a linear stochastic model (A, C, P, Q, R, S) that generates Λ.*

Problem 3.8 (minimal linear stochastic realization problem)**.** *Given autocovariances $\Lambda : \mathbb{Z}_+ \mapsto \mathbb{R}^{p\times p}$. Find a linear stochastic model (A, C, P, Q, R, S), with A as small as possible, that generates Λ.*

An implicit subproblem of the linear stochastic realization problem is the *linear stochastic realizability question*: derive conditions for autocovariances Λ to be representable by a linear stochastic model of finite order. A solution to the linear stochastic realization problem is called a *realization* of the autocovariances Λ. We now define equivalence and minimality of linear stochastic models.

Definition 3.8. *Two linear stochastic models with autocovariances Λ and Λ' respectively, are said to be* equivalent *if $\Lambda = \Lambda'$.*

Definition 3.9. *A linear stochastic model (A, C, P, Q, R, S) of order n is called* minimal *if for any other equivalent linear stochastic model (A', C', P', Q', R', S') of order n' it holds that $n \le n'$.*

Definition 3.10. *A Moore linear stochastic model $(A, C, P, Q, R, 0)$ of order n is called* minimal as a Mealy model *if for any equivalent Mealy linear stochastic model (A', C', P', Q', R', S') of order n' it holds that $n \le n'$.*

3.5 Equivalence of linear stochastic models

One can easily see that for Mealy as well as for Moore linear stochastic models, an equivalent model is obtained by changing the basis in the state space as $x \mapsto Tx$, with T nonsingular. The equivalent model is then given by $(TAT^{-1}, CT^{-1}, TPT^\top, TQT^\top, R, TS)$. This state transformation is the analogue of the permutation of the states which is always possible for HMMs.

However, again analogous to the quasi hidden Markov case, there are more equivalent models for Mealy linear stochastic models than only those obtained by applying a similarity transformation (Figure 3.2). Indeed, it can be proven

[46] that for a given A, C, G and $\Lambda(0)$, i.e. for a given autocovariance sequence and given state space basis, every $P = P^\top \succeq 0$ which fullfills

$$\left[\begin{array}{c|c} P - APA^\top & G - APC^\top \\ \hline G^\top - CPA^\top & \Lambda(0) - CPC^\top \end{array}\right] \succeq 0,$$

where $X \succeq 0$ means that X is nonnegative definite, gives rise to an equivalent model $(A, C, P, P - APA^\top, \Lambda(0) - CPC^\top, G - APC^\top)$. This observation is the analogue of the fact that for Mealy HMMs one has equivalent models which are not obtained by permuting states.

For Moore HMMs on the other hand, under certain conditions, there exist only trivial equivalent models (Theorem 3.2). We here prove the analogous theorem for linear stochastic models.

Theorem 3.4. *Let $(A, C, Q, P, R, 0)$ be a Moore linear stochastic model which is minimal as a Mealy model. Suppose that A has full rank and C has full column rank. Then every Moore model that is equivalent to the given Moore model is obtained by performing a change of basis in the state space.*

Proof: From the fact that $S = 0$ we find that $G - APC^\top = 0$, and from the fact that C has full column rank and A full rank, we find that $P = A^{-1}G(C^\top)^\dagger$. So for a given state space basis there is only one possible choice of P, which proves the theorem. ■

The condition that C has full column rank (condition of Theorem 3.4) is analogous to the condition that a different state at two time instants gives a different output distribution at these time instants. This corresponds to the condition for HMMs which requires every state to have a different output distribution (Theorem 3.2). The fact that A needs to have full rank is the analogue of the fact that for HMMs $\Pi_{\mathbb{X}}$ needs to have full rank. We conclude that Theorem 3.4 is the linear stochastic equivalent of Theorem 3.2.

If C is not of full column rank, but all other conditions of Theorem 3.4 are fulfilled, then there exists a set of equivalent Moore models (apart from the models obtained by performing a base change in the state space).

Theorem 3.5. *Let $(A, C, P, Q, R, 0)$ be a minimal Moore linear stochastic model which is minimal as a Mealy model, with $G = APC^\top$ and $\Lambda(0) = CPC^\top + R$. Suppose that A has full rank and C is not of full column rank. Then the model $(A, C, P', Q', R', 0)$ is an equivalent Moore model if and only if*

$$\begin{aligned} G &= AP'C^\top, \\ Q' &= P' - AP'A^\top, \\ R' &= \Lambda(0) - CP'C^\top. \end{aligned}$$

Proof: The theorem follows from the proof of Theorem 3.4. ■

Notice that Theorem 3.5 is the linear stochastic equivalent of Theorem 3.3. In case the Moore model is not minimal as a Mealy model model, then again analogous to the HMM case (cfr. Proposition 3.5), a complete set of equivalent models exists. We conclude that for linear stochastic model, we have a completely analogous situation as is described in Figure 3.4 for HMMs.

3.6 Conclusions

In this chapter we formally introduced hidden Markov models and linear stochastic models. Concerning hidden Markov models, we provide a test to check whether a quasi Mealy hidden Markov model is minimal and describe a procedure to obtain a minimal quasi Mealy model equivalent to a given nonminimal quasi or positive hidden Markov model.

We also consider the equivalence problem for hidden Markov models. We provide a test to check whether two positive Mealy hidden Markov models are equivalent and give a description of the complete set of equivalent models. Subsequently, we prove that Moore models that are minimal as a quasi Mealy models, under certain conditions, have only trivial equivalents. Moore models that are not minimal as a quasi model on the other hand can have equivalent models. We provide a test for checking the equivalence of Moore models and describe the complete set of equivalent models.

Finally, the equivalence problem for linear stochastic models is considered. It turns out that the situation for linear stochastic models is analogous to the situation for hidden Markov models.

Chapter 4

Quasi realization of hidden Markov models

The quasi realization problem for hidden Markov models consists in finding a quasi hidden Markov model corresponding to given string probabilities of all finite length output strings. The quasi realization problem was first stated in [52, 84]. The quasi realization question consists of three subquestions. The first question is the realizability question: under which conditions are string probabilities representable by a hidden Markov model. The solution to the realizability question lies in the construction of a doubly infinite Hankel matrix containing the string probabilities. It can be shown that string probabilities are realizable by a quasi hidden Markov model if and only if the associated Hankel matrix of string probabilities has finite rank [52, 84]. The second question is the realization question itself. In [4, 112, 113] algorithms are provided to solve the quasi realization problem for hidden Markov models. The starting point of these algorithms is the factorization of the Hankel matrix containing the string probabilities. The third question is the equivalence question find *all* quasi hidden Markov models that realize given string probabilities. The equivalence problem for quasi hidden Markov models has been considered in [112, 113].

Altough the *exact quasi realization problem* is nice from a theoretical point of view, it can not be used in practical applications. The reason therefore is twofold: first of all in practical applications only a finite amount of string probabilities are given instead of string probabilities of all finite length strings (i.e. an infinite amount of string probabilities) and second the given string probabilities may not be exact but only estimated. We introduce the *partial quasi realization problem* that builds a quasi HMM corresponding to a finite amount of exact string probabilities and the *approximate partial quasi realization problem* that assumes a finite amount of approximate string probabilities to be given. To the best of our knowledge the partial and approximate partial quasi realization problems have not been considered before.

The system matrices of a quasi realization do not have an interpretation in terms of probabilities, nevertheless quasi realizations do have their importance in practical applications. It will be shown in Chapter 7, that for several output estimation problems it suffices to have a quasi instead of a positive realization. The advantage of this is twofold. First of all a quasi realization can be obtained more easily as compared to a positive realization. In addition, the order of a quasi realization is typically smaller than the order of an equivalent positive realization which makes the estimation problem less complex.

The quasi realization problem for hidden Markov models is closely related to the realization problem of formal power series [17, 70, 82], which is itself a generalization of realization theory for linear time-invariant deterministic systems [56, 71, 93, 118]. For this last class of systems the partial realization problem has been considered in [54, 63, 98] and the approximate partial realization problem in [71].

List of own contributions

We here describe our contributions to the quasi realization problem for hidden Markov models.

- In Section 4.1.1 we prove that the rank of the finite generalized Hankel matrix of string probabilities is equal to the rank of its last block column (Proposition 4.1) and that the rank of the finite Hankel matrix of stationary string probabilities is equal to the rank of its largest subblock (Proposition 4.2). In Proposition 4.3 we prove that the rank of the Hankel matrix of string probabilities generated by a quasi HMM of order $|\mathbb{Q}|$ is contained in the leading submatrix of blocksize $|\mathbb{Q}| \times |\mathbb{Q}|$.

- In Theorem 4.1 we show that the finite rank property of the generalized Hankel matrix is equivalent to the existence of a recursion between the string probabilities. Algorithm 4.1 provides a way to compute a quasi realization corresponding to given string probabilities. We show further that a quasi realization of consistent string probabilities is consistent and a quasi realization of consistent stationary string probabilities is consistent and stationary.

- In Section 4.2.1 we introduce the partial quasi realization problem, we prove that it always has a solution and provide an algorithm to find the solution. However, the minimal partial realization problem is hard in practice. We introduce the minimal partial pseudo realization problem. We prove that, if a certain rank condition holds, the minimal pseudo realization algorithm can be solved using the same algorithm as was used for the complete quasi realization problem. In addition under the same rank condition, we prove that a solution to the partial pseudo realization problem is unique up to a similarity transformation. We also give some hints for the solution to the partial pseudo relization in case the rank condition does not hold.

- In Section 4.3 we introduce the approximate partial pseudo realization problem for hidden Markov models. Different methods are presented to solve this problem. The first methods aim at finding a low rank approximation of the finite Hankel matrix either by projecting the string probabilities contained in the Hankel matrix on the consistency and/or stationarity constraints or by using low rank matrix approximation techniques or by a combination of both. The last method aims at obtaining a full-order balanced realization of the approximate string probabilities and subsequently reduces the balanced realization to find an approximate pseudo realization of the approximate string probabilities.
- In Section 4.4 the approximate partial quasi realization algorithm is succesfully applied to the problem of modeling DNA sequences. Therefore, first strings probabilities are estimated from the sequences and subsequently the approximate quasi realization algorithm is applied to find a model of the sequences.

Section-by-section overview

In Section 4.1 the exact quasi realization problem is reviewed. In Section 4.2 we consider the partial quasi realization problem and in Section 4.2 the approximate partial quasi realization problem. In Section 4.4 the approximate quasi realization algorithm is applied to the modeling of DNA sequences. In Section 4.5 we compare the quasi realization problem with the realization problem for linear stochastic models.

4.1 Exact quasi realization

In this section we consider the exact minimal quasi realization problem for Mealy hidden Markov models (Problem 3.4). In this problem we are given *exact* string probabilities of a finite-valued process and the problem is to find a quasi Mealy HMM that realizes these string probabilities.

In Section 4.1.1 we introduce the generalized Hankel matrix associated with the string probabilities. This matrix plays a crucial role in the realization problem. In Section 4.1.2 we solve the realizability question and provide an algorithm to find a quasi realization of given string probabilities. In Section 4.1.3 finally, we prove some interesting properties of the obtained quasi realization and relate these properties to properties of the string probabilities.

4.1.1 Generalized Hankel matrix

Define the *(generalized) Hankel matrix* $\mathfrak{H}_{\mathcal{P}}$ of string probabilities $\mathcal{P}$ as a doubly infinite matrix with

$$(\mathfrak{H}_{\mathcal{P}})_{ij} := \mathcal{P}(\mathbf{u}_i\mathbf{v}_j),$$

where $\mathbf{u}_i$ is the i-th element of the set of strings from $\mathbb{Y}^*$ in first lexicographical ordering $\mathcal{U}$ and $\mathbf{v}_j$ is the j-th element of the set of strings from $\mathbb{Y}^*$ in last lexicographical ordering $\mathcal{V}$. We use "$\mathfrak{H}$" instead of "$\mathfrak{H}_\mathcal{P}$" whenever the string probabilities $\mathcal{P}$ that define the Hankel matrix are clear from the context. For refering to the i-th row of $\mathfrak{H}$, we use "the row of $\mathfrak{H}$ indexed by $\mathbf{u}_i$" and for refering to the j-th column of $\mathfrak{H}$, we use "the column of $\mathfrak{H}$ indexed by $\mathbf{v}_j$". In the case where $\mathbb{Y} = \{0,1\}$ the (generalized) Hankel matrix is given by

$$\mathfrak{H} = \left[\begin{array}{c|cc|cccc|c}
1 & \mathcal{P}(0) & \mathcal{P}(1) & \mathcal{P}(00) & \mathcal{P}(01) & \mathcal{P}(10) & \mathcal{P}(11) & \dots \\ \hline
\mathcal{P}(0) & \mathcal{P}(00) & \mathcal{P}(01) & \mathcal{P}(000) & \mathcal{P}(001) & \mathcal{P}(010) & \mathcal{P}(011) & \dots \\
\mathcal{P}(1) & \mathcal{P}(10) & \mathcal{P}(11) & \mathcal{P}(100) & \mathcal{P}(101) & \mathcal{P}(110) & \mathcal{P}(111) & \dots \\ \hline
\mathcal{P}(00) & \mathcal{P}(000) & \mathcal{P}(001) & \mathcal{P}(0000) & \mathcal{P}(0001) & \mathcal{P}(0010) & \mathcal{P}(0011) & \dots \\
\mathcal{P}(10) & \mathcal{P}(100) & \mathcal{P}(101) & \mathcal{P}(1000) & \mathcal{P}(1001) & \mathcal{P}(1010) & \mathcal{P}(1011) & \dots \\
\mathcal{P}(01) & \mathcal{P}(010) & \mathcal{P}(011) & \mathcal{P}(0100) & \mathcal{P}(0101) & \mathcal{P}(0110) & \mathcal{P}(0111) & \dots \\
\mathcal{P}(11) & \mathcal{P}(110) & \mathcal{P}(111) & \mathcal{P}(1100) & \mathcal{P}(1101) & \mathcal{P}(1110) & \mathcal{P}(1111) & \dots \\ \hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}\right].$$

The matrix $\mathfrak{H}_{(t_1,t_2)}$ is defined as the subblock of $\mathfrak{H}$ given by

$$\mathfrak{H}_{(t_1,t_2)} := [\mathcal{P}(\mathbf{u}_i\mathbf{v}_j)], \text{ with } |\mathbf{u}_i| = t_1 - 1, |\mathbf{v}_j| = t_2 - 1.$$

The name "generalized Hankel" matrix comes from the analogy with the block Hankel matrix that is frequently used in system theory of linear time-invariant systems. In a block Hankel matrix, the blocks along the anti-diagonals are equal to each other. The generalized Hankel matrix is not of the block Hankel structure. However, it has similar properties. First of all, the blocks along every anti-diagonal contain string probabilities of strings of the same length. For instance, the blocks $\mathfrak{H}_{(3,1)}$, $\mathfrak{H}_{(2,2)}$ and $\mathfrak{H}_{(1,3)}$ all contain string probabilities of strings of length 2. Moreover, the blocks along a certain anti-diagonal can be constructed from each other. We explain this for the case where $|\mathbb{Y}| = 2$. Given the block $\mathfrak{H}_{(1,3)}$, *cut* it in the middle and *put* the right hand part below the left hand part. This gives the block $\mathfrak{H}_{(2,2)}$. To find the block $\mathfrak{H}_{(3,1)}$, the procedure is similar: *cut* the block $\mathfrak{H}_{(2,2)}$ in the middle and *put* the right hand part below the left hand part. This gives $\mathfrak{H}_{(3,1)}$.

The *finite Hankel matrix* $\mathfrak{H}_{(1:t_1,1:t_2)}$ is defined as the top-left corner of the Hankel matrix $\mathfrak{H}$

$$\begin{aligned}
\mathfrak{H}_{(1:t_1,1:t_2)} &:= [\mathcal{P}(\mathbf{u}_i\mathbf{v}_j)], \text{ with } |\mathbf{u}_i| \le t_1 - 1, |\mathbf{v}_j| \le t_2 - 1, \\
&= \begin{bmatrix} \mathfrak{H}_{(1,1)} & \mathfrak{H}_{(1,2)} & \cdots & \mathfrak{H}_{(1,t_2)} \\ \mathfrak{H}_{(2,1)} & \mathfrak{H}_{(2,2)} & \cdots & \mathfrak{H}_{(2,t_2)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathfrak{H}_{(t_1,1)} & \mathfrak{H}_{(t_1,2)} & \cdots & \mathfrak{H}_{(t_1,t_2)} \end{bmatrix}.
\end{aligned}$$

Suppose $\mathcal{M} = (\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_m)$ is an ordered subset of strings from $\mathbb{Y}^*$ in first lexicographical ordering and $\mathcal{N} = (\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_n)$ in last lexicographical ordering. Then the matrix $\mathfrak{H}_{(\mathcal{M},\mathcal{N})}$ is defined as the submatrix of $\mathfrak{H}$ given by

$$(\mathfrak{H}_{(\mathcal{M},\mathcal{N})})_{ij} = \mathcal{P}(\mathbf{m}_i\mathbf{n}_j).$$

We now prove that the rank of the finite Hankel matrix $\mathfrak{H}_{(1:t_1,1:t_2)}$ is equal to the rank of its last block-column. In case the string probabilities are stationary, the rank of the finite Hankel matrix $\mathfrak{H}_{(1:t_1,1:t_2)}$ is equal to the rank of the largest subblock.

Proposition 4.1. *Consider the finite Hankel matrix $\mathfrak{H}_{(1:t_1,1:t_2)}$ of string probabilities $\mathcal{P}$. There holds*

$$\operatorname{rank}\mathfrak{H}_{(1:t_1,1:t_2)} = \operatorname{rank}\mathfrak{H}_{(1:t_1,t_2)}.$$

Proof: We first prove that $\operatorname{rank}\mathfrak{H}_{(1:t_1,t_2-1:t_2)} = \operatorname{rank}\mathfrak{H}_{(1:t_1,t_2)}$. This can be seen from the fact that the column of $\mathfrak{H}_{(1:t_1,t_2-1)}$ indexed by a string $\mathbf{u}$ is equal to the sum of the $|\mathbb{Y}|$ columns of $\mathfrak{H}_{(1:t_1,t_2)}$ indexed by $\mathbf{uy}, \mathrm{y} \in \mathbb{Y}$. This follows immediately from the consistency of the string probabilities: $\sum_{\mathrm{y}\in\mathbb{Y}} \mathcal{P}(\mathbf{vy}) = \mathcal{P}(\mathbf{v})$. By induction, it can be proven that $\operatorname{rank}\mathfrak{H}_{(1:t_1,t_2)} = \operatorname{rank}\mathfrak{H}_{(1:t_1,t_2-1:t_2)} = \operatorname{rank}\mathfrak{H}_{(1:t_1,t_2-2:t_2)} = \ldots = \operatorname{rank}\mathfrak{H}_{(1:t_1,1:t_2)}$, which proves the proposition. ■

Proposition 4.2. *Consider the finite Hankel matrix $\mathfrak{H}_{(1:t_1,1:t_2)}$ of stationary string probabilities $\mathcal{P}$. There holds*

$$\operatorname{rank}\mathfrak{H}_{(1:t_1,1:t_2)} = \operatorname{rank}\mathfrak{H}_{(t_1,t_2)}.$$

Proof: The proof is like the proof of Proposition 4.1. ■

If the strings probabilities $\mathcal{P}$ are generated by the quasi model $(\mathbb{Q}, \mathbb{Y}, A, c, b)$, then the corresponding Hankel matrix can be factorized as

$$\mathfrak{H} = \mathcal{O}(c, A)\mathcal{C}(A, b).$$

This can be seen from

$$\mathfrak{H} = \left[\begin{array}{c|cc|cccc|c}
cb & cA(0)b & cA(1)b & cA(00)b & cA(01)b & cA(10)b & cA(11)b & \ldots \\ \hline
cA(0)b & cA(00)b & cA(01)b & cA(000)b & cA(001)b & cA(010)b & cA(011)b & \ldots \\
cA(1)b & cA(10)b & cA(11)b & cA(100)b & cA(101)b & cA(110)b & cA(111)b & \ldots \\ \hline
cA(00)b & cA(000)b & cA(001)b & cA(0000)b & cA(0001)b & cA(0010)b & cA(0011)b & \ldots \\
cA(10)b & cA(100)b & cA(101)b & cA(1000)b & cA(1001)b & cA(1010)b & cA(1011)b & \ldots \\
cA(01)b & cA(010)b & cA(011)b & cA(0100)b & cA(0101)b & cA(0110)b & cA(0111)b & \ldots \\
cA(11)b & cA(110)b & cA(111)b & cA(1100)b & cA(1101)b & cA(1110)b & cA(1111)b & \ldots \\ \hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}\right]$$

$$= \left[\begin{array}{c}
c \\ \hline
cA(0) \\
cA(1) \\ \hline
cA(00) \\
cA(10) \\
cA(01) \\
cA(11) \\ \hline
\vdots
\end{array}\right]
\left[\begin{array}{c|cc|cccc|c}
b & A(0)b & A(1)b & A(00)b & A(01)b & A(10)b & A(11)b & \ldots
\end{array}\right].$$

Now, we can prove the following proposition.

Proposition 4.3. *Consider a Hankel matrix $\mathfrak{H}$ of string probabilities generated by a quasi Mealy model of order $|\mathbb{Q}|$. There holds*

$$\operatorname{rank}\mathfrak{H} = \operatorname{rank}\mathfrak{H}_{(1:|\mathbb{Q}|+l,1:|\mathbb{Q}|+m)}, \quad l = 0, 1, 2, \ldots; m = 0, 1, 2, \ldots \tag{4.1}$$

Proof: Define r as the rank of $\mathfrak{H}_{(1:|\mathbb{Q}|,1:|\mathbb{Q}|)}$. Note that $r \leq |\mathbb{Q}|$. The matrix $\mathfrak{H}_{(1:|\mathbb{Q}|,1:|\mathbb{Q}|)}$ can now be decomposed as $\mathcal{O}_{(1:|\mathbb{Q}|)}(c,A)\mathcal{C}_{(1:|\mathbb{Q}|)}(A,b)$ where $\mathcal{O}_{(1:|\mathbb{Q}|)}(c,A)$ and $\mathcal{C}_{(1:|\mathbb{Q}|)}(A,b)$ have rank r. Now it follows from Proposition 3.1 that $\operatorname{rank}\mathcal{O}_{(1:|\mathbb{Q}|+l)}(c,A) = \operatorname{rank}\mathcal{O}(c,A)$ for $l = 0,1,2,\ldots$ and that $\operatorname{rank}\mathcal{C}_{(1:|\mathbb{Q}|+m)}(A,b) = \operatorname{rank}\mathcal{C}(A,b)$ for $m = 0,1,2,\ldots$, which proves the proposition. ■

It is possible that there exists a $k < |\mathbb{Q}|$ so that $\operatorname{rank}\mathfrak{H} = \operatorname{rank}\mathfrak{H}_{(1:k+l,1:k+m)}$, for $l = 0,1,2,\ldots$ and $m = 0,1,2,\ldots$. However, we do not go into detail about this.

Equation (4.1) can be rewritten as $\operatorname{rank}\mathfrak{H} = \operatorname{rank}\mathfrak{H}_{(1:|\mathbb{Q}|+l,|\mathbb{Q}|+m)}$, for $l = 0,1,2,\ldots$ and $m = 0,1,2,\ldots$, because of Proposition 4.1. In case the string probabilities are stationary, Proposition 4.2 allows to rewrite (4.1) as $\operatorname{rank}\mathfrak{H} = \operatorname{rank}\mathfrak{H}_{(|\mathbb{Q}|+l,|\mathbb{Q}|+m)}$, for $l = 0,1,2,\ldots$ and $m = 0,1,2,\ldots$.

4.1.2 Exact quasi realization

In this section we consider the exact quasi realization problem and the minimal exact quasi realization problem (Problem 3.3 and Problem 3.4). We first present an answer to the quasi Mealy realizability question: under which conditions are string probabilities $\mathcal{P}$ representable by a quasi Mealy HMM of finite order. Next, we describe how to find the minimal order of a realization of string probabilities and finally, we present a quasi realization algorithm.

The quasi Mealy realizability question is solved by the following theorem.

Theorem 4.1. *The following are equivalent*

(1) String probabilities $\mathcal{P}$ are realizable by a quasi Mealy HMM $(\mathbb{Q},\mathbb{Y},A,c,b)$ with $|\mathbb{Q}| < \infty$.

(2) The rank of the generalized Hankel matrix $\mathfrak{H}$ of string probabilities $\mathcal{P}$ is finite.

(3) There exist a $k \in \mathbb{N}$ and constants $\alpha_{\mathbf{v},\mathbf{u}} \in \mathbb{R}, \mathbf{u} \in \mathbb{Y}^k, \mathbf{v} \in \mathbb{Y}^{k-1}$ such that the string probabilities $\mathcal{P}$ satisfy

$$\mathcal{P}(\mathbf{wu}) = \sum_{\mathbf{v}\in\mathbb{Y}^{k-1}} \alpha_{\mathbf{v},\mathbf{u}}\mathcal{P}(\mathbf{wv}), \tag{4.2}$$

for all $\mathbf{u} \in \mathbb{Y}^k$ and for all $\mathbf{w} \in \mathbb{Y}^*$.

(4) There exist a $l \in \mathbb{N}$ and constants $\beta_{\mathbf{v},\mathbf{u}} \in \mathbb{R}, \mathbf{u} \in \mathbb{Y}^l, \mathbf{v} \in \mathbb{Y}^*, |\mathbf{v}| < l$ such that the string probabilities $\mathcal{P}$ satisfy

$$\mathcal{P}(\mathbf{uw}) = \sum_{\mathbf{v}\in\mathbb{Y}^*,|\mathbf{v}|<l} \beta_{\mathbf{v},\mathbf{u}}\mathcal{P}(\mathbf{vw}), \tag{4.3}$$

for all $\mathbf{u} \in \mathbb{Y}^l$ and for all $\mathbf{w} \in \mathbb{Y}^*$.

Proof: The equivalence of (1) and (2) is proven in [4, 112, 113]. We here first prove the equivalence of (1)-(2) and (3).

From (4.2) it follows that $\operatorname{rank} \mathfrak{H}_{(1:\infty,1:k)} = \operatorname{rank} \mathfrak{H}_{(1:\infty,1:k+1)}$. Now by taking $\mathbf{w} = \mathbf{mn}$ with $|\mathbf{n}| = 1$, it follows from (4.2) that $\operatorname{rank} \mathfrak{H}_{(1:\infty,1:k+1)} = \operatorname{rank} \mathfrak{H}_{(1:\infty,1:k+2)}$. By continuing with $|\mathbf{n}| = 2, 3, \ldots$, it is proven that $\operatorname{rank} \mathfrak{H}_{(1:\infty,1:\infty)} = \operatorname{rank} \mathfrak{H}_{(1:\infty,1:k)} = \operatorname{rank} \mathfrak{H}_{(1:\infty,k)}$, where the last equality follows from Proposition 4.1. It follows that $\operatorname{rank} \mathfrak{H}_{(1:\infty,1:\infty)}$ is finite.

On the other hand from the existence of a realization of order $|\mathbb{Q}| < \infty$ and from Proposition 4.3, it follows that $\operatorname{rank} \mathfrak{H} = \operatorname{rank} \mathfrak{H}_{(1:|\mathbb{Q}|+l,|\mathbb{Q}|+m)}$, $l = 0, 1, 2, \ldots; m = 0, 1, 2, \ldots$. Hence for $k = |\mathbb{Q}|$, there exist constants $\alpha_{\mathbf{v},\mathbf{u}}, \mathbf{u} \in \mathbb{Y}^k, \mathbf{v} \in \mathbb{Y}^{k-1}$ such that (4.2) holds.

The equivalence of (1)-(2) and (4) is proven in a analogous way with the only difference that for nonstationary string probabilities, we do not have that $\operatorname{rank} \mathfrak{H}_{(1:l,1:\infty)} = \operatorname{rank} \mathfrak{H}_{(l,1:\infty)}$. For that reason, we sum over $\mathbf{v} \in \mathbb{Y}^*, |\mathbf{v}| < l$ in (4.3) instead of over $\mathbf{v} \in \mathbb{Y}^{l-1}$. ■

The following theorem provides a way to find the minimal order of a quasi realization of string probabilities $\mathcal{P}$ [4, 112, 113].

Theorem 4.2. *The order of a minimal quasi realization of string probabilities $\mathcal{P}$ is equal to the rank of the Hankel matrix $\mathfrak{H}_{\mathcal{P}}$.*

Any two solutions to the minimal quasi realization problem are connected by a similarity transformation as described in Proposition 3.2.

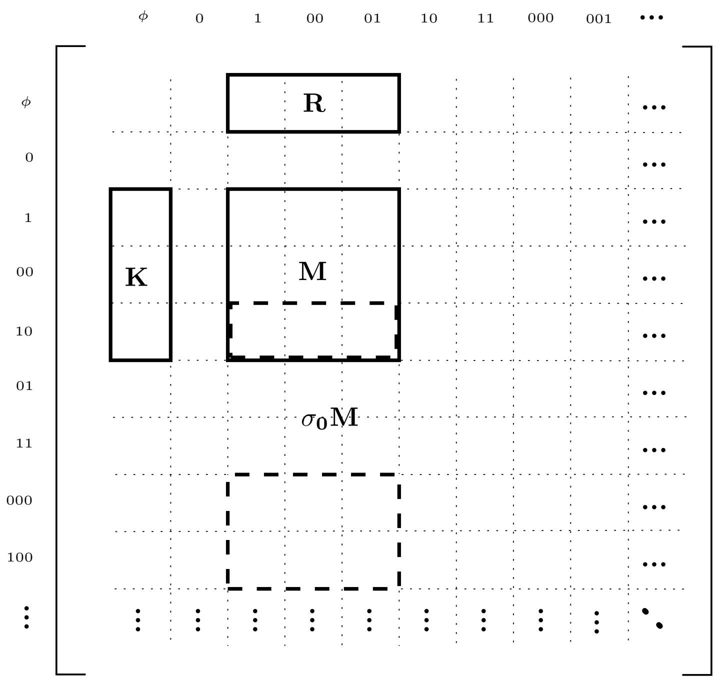
In [4], an algorithm is presented to find a minimal quasi realization for a given finite rank Hankel matrix. We here present a more general algorithm to solve the minimal quasi realization problem. In Proposition 4.4, we prove the correctness of the algorithm.

Algorithm 4.1. *Given a rank r Hankel matrix $\mathfrak{H}$ ($r < \infty$) of string probabilities $\mathcal{P}$ of a process taking values in the finite set $\mathbb{Y}$. Perform the following steps.*

1. *Choose $\mathcal{M} = (\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_m)$ as an ordered subset of strings from $\mathbb{Y}^*$ in first lexicographical ordering and $\mathcal{N} = (\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_n)$ in last lexicographical ordering such that* $\operatorname{rank} M = \operatorname{rank} \mathfrak{H}$ *where $M := \mathfrak{H}_{(\mathcal{M},\mathcal{N})}$.*
2. *Define $R := \mathfrak{H}_{(\phi,\mathcal{N})}$ and $K := \mathfrak{H}_{(\mathcal{M},\phi)}$, where ϕ is the empty string. For each $y \in \mathbb{Y}$ define $\sigma_y M$ as $\mathfrak{H}_{(\sigma_y\mathcal{M},\mathcal{N})}$, where $\sigma_y\mathcal{M} := (\mathbf{m}_1 y, \mathbf{m}_2 y, \ldots, \mathbf{m}_m y)$. Equivalently, by the Hankel structure, $\sigma_y M$ can be defined as $\mathfrak{H}_{(\mathcal{M},\sigma_y\mathcal{N})}$, where $\sigma_y\mathcal{N} := (y\mathbf{n}_1, y\mathbf{n}_2, \ldots, y\mathbf{n}_n)$.*
3. *Find $P \in \mathbb{R}^{r\times m}$ and $Q \in \mathbb{R}^{n\times r}$ such that $PMQ = I_r$.*
4. *A minimal quasi realization $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ is now obtained as follows:*

$$\begin{aligned} \mathbb{Q} &= \{1, 2, \ldots, r\}, \\ A(y) &= P\sigma_y MQ, \quad \forall\, y \in \mathbb{Y}, \\ c &= RQ, \\ b &= PK. \end{aligned}$$

In Figure 4.1, we illustrate the definition of the different matrices of Algorithm 4.1 for the case where $\mathbb{Y} = \{0,1\}$ and $r = 3$. We suppose that $M := \mathfrak{H}_{((1,00,10),(1,00,01))}$ has rank 3. Then $\sigma_0 M$ is given by $\sigma_0 M = \mathfrak{H}_{((10,000,100),(1,00,01))}$ as indicated in the figure. The matrix $\sigma_1 M$ is defined analogously.



ϕ 0 1 00 01 10 11 000 001
ϕ 0 1 00 10 01 11 000 100
R
K
M
σ_0M

Figure 4.1: *Definition of the different matrices of Algorithm 4.1 for the case where* $\mathbb{Y} = \{0,1\}$ *and* $r = 3$.

We now prove the correctness of Algorithm 4.1.

Proposition 4.4. *Algorithm 4.1 yields a solution to the minimal quasi realization problem of string probabilities* $\mathcal{P}$.

Proof: Define the infinite selector matrices

$$\begin{aligned} S_R &= \begin{bmatrix} s'_1 \\ s'_2 \\ \vdots \\ s'_m \end{bmatrix}, \\ S_K &= \begin{bmatrix} s''_1 & s''_2 & \dots & s''_n \end{bmatrix}, \end{aligned}$$

where $s'_i = [0,\ldots,0,1,0,\ldots]$ where the 1 is at the r_i-th position where r_i is the position of string $\mathbf{m}_i$ in the first lexicographical ordering $\mathcal{U}$ and where $s''_i = [0,\ldots,0,1,0,\ldots]^\top$ where the 1 is at the k_i-th position where k_i is the position of string $\mathbf{n}_i$ in the last lexicographical ordering $\mathcal{V}$. Then $M = S_R\mathfrak{H}S_K$, $\sigma_{\mathrm{y}}M = S_R\sigma_{\mathrm{y}}\mathfrak{H}S_K$, $R = S_R\mathfrak{H}_{:,1}$, $K = \mathfrak{H}_{1,:}S_K$. It follows that

$$\begin{aligned} PS_R\mathfrak{H}S_KQ &= I_r, \\ PS_R\sigma_{\mathrm{y}}\mathfrak{H}S_KQ &= A(\mathrm{y}), \\ S_R\mathfrak{H}_{:,1}Q &= c, \\ P\mathfrak{H}_{1,:}S_K &= b. \end{aligned}$$

Now assume that $(\bar{\mathbb{Q}},\mathbb{Y},\bar{A},\bar{c},\bar{b})$ is a minimal quasi realization of the string probabilities contained in $\mathfrak{H}$. Then

$$\mathfrak{H} = \left[\begin{array}{c} \bar{c} \\ \hline \bar{c}\bar{A}(0) \\ \bar{c}\bar{A}(1) \\ \hline \bar{c}\bar{A}(00) \\ \bar{c}\bar{A}(10) \\ \bar{c}\bar{A}(01) \\ \bar{c}\bar{A}(11) \\ \hline \vdots \end{array}\right] \left[\begin{array}{c|cc|cccc|c} \bar{b} & \bar{A}(0)\bar{b} & \bar{A}(1)\bar{b} & \bar{A}(00)\bar{b} & \bar{A}(01)\bar{b} & \bar{A}(10)\bar{b} & \bar{A}(11)\bar{b} & \ldots \end{array}\right].$$

Define

$$T := PS_R \left[\begin{array}{c} \bar{c} \\ \hline \bar{c}\bar{A}(0) \\ \bar{c}\bar{A}(1) \\ \hline \bar{c}\bar{A}(00) \\ \bar{c}\bar{A}(10) \\ \bar{c}\bar{A}(01) \\ \bar{c}\bar{A}(11) \\ \hline \vdots \end{array}\right].$$

Then $PS_R\mathfrak{H}S_KQ = I_r$ implies that

$$\left[\begin{array}{c|cc|cccc|c} \bar{b} & \bar{A}(0)\bar{b} & \bar{A}(1)\bar{b} & \bar{A}(00)\bar{b} & \bar{A}(01)\bar{b} & \bar{A}(10)\bar{b} & \bar{A}(11)\bar{b} & \ldots \end{array}\right] S_KQ = T^{-1}.$$

Now it can be easily verified that

$$\begin{aligned} PS_R\sigma_{\mathrm{y}}\mathfrak{H}S_KQ = A(\mathrm{y}) \quad &\Rightarrow \quad T\bar{A}T^{-1} = A, \\ S_R\mathfrak{H}_{:,1}Q = c \quad &\Rightarrow \quad \bar{c}T^{-1} = c, \\ P\mathfrak{H}_{1,:}S_K = b \quad &\Rightarrow \quad T\bar{b} = b. \end{aligned}$$

From Proposition 3.2, it follows that $(\mathbb{Q},\mathbb{Y},A,c,b)$ is a minimal quasi realization of the string probabilities, which proves the proposition. ■

The solution to the minimal quasi realization problem is not unique. It can be proven that any two different solutions are connected by a similarity transformation. We here do not go into detail about this equivalence question, as it was already considered in the previous chapter (Proposition 3.2).

4.1.3 Properties of the obtained quasi realization

In this section we prove some interesting properties of quasi realizations of string probabilities and relate them to properties of the corresponding string probabilities.

Proposition 4.5. *Consider a quasi realization* $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ *of consistent string probabilities* $\mathcal{P}$*. If* $\mathcal{C}(c, A)$ *has full column rank, then* $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ *is consistent (i.e.* $\sum_{y \in \mathbb{Y}} \Pi(y) b = b$ *and* $cb = 1$*).*

Proof: For stationary string probabilities it holds that $\sum_{\mathrm{y} \in \mathbb{Y}} \mathcal{P}(\mathbf{u}\mathrm{y}) = \mathcal{P}(\mathbf{u}), \forall \mathbf{u} \in \mathbb{Y}^*$. It follows that $cA(\mathbf{u}) \sum_{\mathrm{y} \in \mathbb{Y}} A(\mathrm{y}) b = cA(\mathbf{u}) b$, $\forall \mathbf{u} \in \mathbb{Y}^*$. Since $\mathcal{C}(c, A)$ has full column rank, it follows that $\sum_{\mathrm{y} \in \mathbb{Y}} A(\mathrm{y}) b = b$. On the other hand, from $\sum_{\mathbf{u} \in \mathbb{Y}^t} \mathcal{P}(\mathbf{u}) = 1$, it follows that $c(\sum_{\mathrm{y} \in \mathbb{Y}} A(\mathrm{y}))^t b = 1$. Combining both results gives $cb = 1$.

■

Proposition 4.6. *Consider a quasi realization* $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ *of stationary string probabilities* $\mathcal{P}$*. If* $\mathcal{O}(A, b)$ *has full row rank, then* $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ *is stationary (i.e.* $c \sum_{y \in \mathbb{Y}} \Pi(y) = c$*).*

Proof: For stationary string probabilities it holds that $\sum_{\mathrm{y} \in \mathbb{Y}} \mathcal{P}(\mathrm{y}\mathbf{u}) = \mathcal{P}(\mathbf{u}), \forall \mathbf{u} \in \mathbb{Y}^*$. It follows that $c \sum_{\mathrm{y} \in \mathbb{Y}} A(\mathrm{y}) A(\mathbf{u}) b = cA(\mathbf{u}) b$, $\forall \mathbf{u} \in \mathbb{Y}^*$. Since $\mathcal{O}(A, b)$ has full row rank, it follows that $c \sum_{\mathrm{y} \in \mathbb{Y}} A(\mathrm{y}) = c$. ■

In [4, 112, 113] it is proven that a process with a certain type of *long-term independence* gives rise to a quasi hidden Markov model with spectral radius one. In words, long term independence means that two parts in the process that are far away from each other, can be considered independent. String probabilities of a process with long-term independence, are called *alpha-mixing.* We now formally review these results.

Definition 4.1. *Stationary string probabilities* $\mathcal{P}$ *are called* alpha-mixing *if it holds for every* $\mathbf{u} \in \mathbb{Y}^*$ *and* $\mathbf{v} \in \mathbb{Y}^*$ *that*

$$\sum_{\mathbf{w} \in \mathbb{Y}^k} \mathcal{P}(\mathbf{uwv}) \to \mathcal{P}(\mathbf{u}) \mathcal{P}(\mathbf{v}) \text{ as } \quad k \to \infty. \tag{4.4}$$

Proposition 4.7. *Consider a quasi realization* $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ *of alpha-mixing string probabilities* $\mathcal{P}$*. Then* $A_{\mathbb{Q}} := \sum_{y \in \mathbb{Y}} A(y)$ *has a spectral radius one and the eigenvalue 1 is simple. In addition it holds that* $A_{\mathbb{Q}}^k \to bc$ *as* $k \to \infty$*.*

4.2 Partial quasi realization

The quasi realization problem as solved in the previous section is interesting from a theoretical point of view. However, it supposes that an infinite number of string probabilities is given, which is not feasible in practical applications. In this section we consider a more realistic situation, where the *exact* string probabilities are given for all strings *up to a certain length* t, and the problem is to find a quasi realization. This problem can be compared to the partial realization problem for linear time-invariant systems as solved in [98].

In Section 4.2.1 we show that the partial quasi realization problem always has a solution and provide a way to obtain it. However, we show that the minimal partial realization problem is hard to solve. In Section 4.2.2 we define pseudo realizations of partial string probabilities and solve the minimal partial pseudo realization problem.

4.2.1 Partial quasi realization

We define *t-partial string probabilities* of a process y as $\mathcal{P}^{(t)} : \{\mathbf{u} \in \mathbb{Y}^*, |\mathbf{u}| \leq t\} \mapsto [0,1]$ as

$$\mathcal{P}^{(t)}(\mathbf{u}) := P(y(1) = u_1, y(2) = u_2, \ldots, \mathrm{y}(|\mathbf{u}|) = u_{|\mathbf{u}|}).$$

In what follows, we use "partial string probabilities" instead of "t-partial string probabilities", if the horizon t is clear from the context. An *extension of partial string probabilities* $\mathcal{P}^{(t)}$ *up to length* T is defined as a mapping $\mathcal{P}^{(T)}_{(t)} : \{\mathbf{u} \in \mathbb{Y}^*, t < |\mathbf{u}| \leq T\} \mapsto [0,1]$. Given partial string probabilities $\mathcal{P}^{(t)}$ and an extension $\mathcal{P}^{(T)}_{(t)}$, the *total string probabilities* of $\mathcal{P}^{(t)}$ and $\mathcal{P}^{(T)}_{(t)}$ are defined as $\mathcal{P}^{(t,T)} : \{\mathbf{u} \in \mathbb{Y}^*, |\mathbf{u}| \leq T\} \mapsto [0,1]$ where

$$\mathcal{P}^{(t,T)}(\mathbf{u}) = \begin{cases} \mathcal{P}^{(t)}(\mathbf{u}), & |\mathbf{u}| \leq t, \\ \mathcal{P}^{(T)}_{(t)}(\mathbf{u}), & t < |\mathbf{u}| \leq T. \end{cases}$$

We now formulate the partial quasi Mealy realization problem and the minimal partial quasi Mealy realization problem.

Problem 4.1 (partial quasi Mealy realization problem)**.** *Given partial string probabilities* $\mathcal{P}^{(t)} : \{\mathbf{u} \in \mathbb{Y}^*, |\mathbf{u}| \leq t\} \mapsto [0,1]$*. Find a quasi Mealy HMM* $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ *that generates* $\mathcal{P}^{(t)}$*.*

Problem 4.2 (minimal partial quasi Mealy realization problem)**.** *Given partial string probabilities* $\mathcal{P}^{(t)} : \{\mathbf{u} \in \mathbb{Y}^*, |\mathbf{u}| \leq t\} \mapsto [0,1]$*. Find a quasi Mealy HMM* $(\mathbb{Q}, \mathbb{Y}, A, c, b)$*, with* $|\mathbb{Q}|$ *as small as possible, that generates* $\mathcal{P}^{(t)}$*.*

A solution to the partial quasi Mealy realization problem with partial string probabilities $\mathcal{P}^{(t)}$ is called a *partial realization* of $\mathcal{P}^{(t)}$. A solution to the minimal partial quasi Mealy realization problem with partial string probabilities $\mathcal{P}^{(t)}$ is called a *minimal partial realization* of $\mathcal{P}^{(t)}$.

We first ask the question whether it is always possible to generate an extension $\mathcal{P}^{(\infty)}_{(t)}$ to the partial string probabilities $\mathcal{P}^{(t)}$ such that the total string probabilities $\mathcal{P}^{(t,\infty)}$ are representable by a quasi Mealy model $(\mathbb{Q}, \mathbb{Y}, A, c, b)$. One might think that chosing arbitrary $\alpha_{\mathbf{v},\mathbf{u}} \in \mathbb{R}, \mathbf{u} \in \mathbb{Y}^{t+1}, \mathbf{v} \in \mathbb{Y}^t$ and calculating the extension $\mathcal{P}^{(\infty)}_{(t)}$ using (4.2) with $k = t+1$, provides the solution. However, the $\alpha_{\mathbf{v},\mathbf{u}}$'s need to be such that

- The total string probabilities $\mathcal{P}^{(t,\infty)}$ are consistent.
- The extension $\mathcal{P}_{(t)}^{(\infty)}$ has range $[0,1]$.

The consistency constraint can be written as $\sum_{\mathbf{y}\in\mathbb{Y}} \mathcal{P}(\mathbf{wuy}) = \mathcal{P}(\mathbf{wu})$ for all $\mathbf{w} \in \mathbb{Y}^*$ and $\mathbf{u} \in \mathbb{Y}^t$. Using (4.2), the consistency constraint can be written as a constraint on the constants $\alpha_{\mathbf{v},\mathbf{uy}}, \mathbf{u} \in \mathbb{Y}^t, \mathbf{v} \in \mathbb{Y}^t$

$$\sum_{\mathbf{y}\in\mathbb{Y}} \sum_{\mathbf{v}\in\mathbb{Y}^t} \alpha_{\mathbf{v},\mathbf{uy}} \mathcal{P}(\mathbf{wv}) = \mathcal{P}(\mathbf{wu}), \quad \forall \mathbf{w} \in \mathbb{Y}^*, \forall \mathbf{u} \in \mathbb{Y}^t. \tag{4.5}$$

By taking

$$\sum_{\mathbf{y}\in\mathbb{Y}} \alpha_{\mathbf{v},\mathbf{uy}} = \begin{cases} 1, & \mathbf{u} = \mathbf{v}, \\ 0, & \mathbf{u} \neq \mathbf{v}, \end{cases} \tag{4.6}$$

(4.5) is fullfilled. Moreover, if $\mathfrak{H}_{(1:\infty,t+1)}$ has full column rank, then (4.6) is the unique solution such that (4.5) is fullfilled.

The constraint that the extension $\mathcal{P}_{(t)}^{(\infty)}$ has range $[0,1]$ can be written as $0 \leq \mathcal{P}(\mathbf{wu}) \leq 1, \mathbf{w} \in \mathbb{Y}^*, \mathbf{u} \in \mathbb{Y}^{t+1}$. Using (4.2), this constraint can be rewritten as a constraint on the constants $\alpha_{\mathbf{v},\mathbf{u}}, \mathbf{u} \in \mathbb{Y}^{t+1}, \mathbf{v} \in \mathbb{Y}^t$

$$0 \leq \sum_{\mathbf{v}\in\mathbb{Y}^t} \alpha_{\mathbf{v},\mathbf{u}} \mathcal{P}(\mathbf{wv}) \leq 1, \quad \mathbf{w} \in \mathbb{Y}^*, \mathbf{u} \in \mathbb{Y}^{t+1}. \tag{4.7}$$

It is hard to check whether a certain choice of $\alpha_{\mathbf{v},\mathbf{u}}, \mathbf{u} \in \mathbb{Y}^{t+1}, \mathbf{v} \in \mathbb{Y}^t$ fullfills (4.7) or not. However, we can prove the following proposition.

Proposition 4.8. *Partial string probabilities $\mathcal{P}^{(t)}$ can be extended with $\mathcal{P}_{(t)}^{(\infty)}$: $\{\mathbf{u} \in \mathbb{Y}^*, |\mathbf{u}| > t\} \mapsto [0,1]$ such that the total strings probabilities $\mathcal{P}^{(t,\infty)}$ are representable by a minimal quasi realization $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ with $|\mathbb{Q}| < \infty$.*

Proof: Take $\alpha_{\mathbf{v},\mathbf{u}}, \mathbf{u} \in \mathbb{Y}^{t+1}, \mathbf{v} \in \mathbb{Y}^t$ that fullfill (4.6) and that are in addition nonnegative such that the extension of the string probabilities is in the interval $[0,1]$. Such choice of $\alpha_{\mathbf{v},\mathbf{uy}}$ is always possible. Now the extension of the string probabilities $\mathcal{P}_{(t)}^{(\infty)}$ can be calculated using Equation (4.2) with $k = t+1$. The Hankel matrix associated with the string probabilities has finite rank, and hence the string probabilities are representable by a minimal quasi HMM (for instance obtained by Algorithm 4.1). ∎

This result proves that a partial quasi realization that is minimal in the sense of Theorem 3.1 always exists. However, it has not yet been shown that a minimal partial quasi realization (a solution to Problem 4.2) always exists. Intuitively, the existence of a minimal partial quasi realization is obvious. Since there always exist partial quasi realizations (Proposition 4.8), there exists at least one which has a smallest dimension.

The minimal partial quasi realization problem is hard to solve in practice. For $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ to be a partial realization of the partial string probabilities $\mathcal{P}^{(t)}$, it is needed that

- $\mathcal{P}^{(t)}(\mathbf{u}) = cA(\mathbf{u})b, \quad \mathbf{u} = u_1u_2\dots u_{|\mathbf{u}|} \in \mathbb{Y}^*, |\mathbf{u}| \leq t.$
- $\mathcal{P}^{(\infty)}_{(t)}(\mathbf{u}) := cA(\mathbf{u})b, \quad |\mathbf{u}| > t$ has range $[0,1]$.
- $\mathcal{P}^{(t,\infty)}$ are consistent string probabilities.

The first and the last constraint are easy to meet. The second constraint however is hard, but not necessary in some applications (see Chapter 7). In the next section we solve the minimal partial pseudo realization problem, i.e. the minimal partial quasi realization problem without the second constraint.

4.2.2 Partial pseudo realization

In this section we consider the minimal partial pseudo realization problem, a relaxed version of the minimal partial quasi realization problem. We first define pseudo realizations and the minimal partial pseudo realization problem. We show further that if a certain rank condition holds, a solution to the minimal partial pseudo realization problem can be obtained using Algorithm 4.1. Furthermore, if the rank condition holds, any two solutions to the minimal partial pseudo realization problem are connected by a similarity transformation. In case the rank condition does not hold, the situation is more complicated. We do not go into full detail about this last situation.

Define a *t-pseudo Mealy HMM* as $(\bar{\mathbb{Q}}, \mathbb{Y}, \bar{A}, \bar{c}, \bar{b})$, where $\bar{\mathbb{Q}}$ is the *pseudo state set* and $\mathbb{Y}$ is the *output set*. The number of states $|\bar{\mathbb{Q}}|$ is called the *order* of the pseudo HMM. $\bar{b}$ is a column vector in $\mathbb{R}^{|\bar{\mathbb{Q}}|}$, $\bar{A}$ is a mapping from $\mathbb{Y}$ to $\mathbb{R}^{|\bar{\mathbb{Q}}|\times|\bar{\mathbb{Q}}|}$, where $\bar{A}_{\bar{\mathbb{Q}}} := \sum_{\mathrm{y}\in\mathbb{Y}} \bar{A}(\mathrm{y})$ is a quasi stochastic matrix, i.e. $\bar{A}_{\mathbb{Q}}\bar{b} = \bar{b}$. The matrices $\bar{A}$, $\bar{c}$, $\bar{b}$ are such that $\bar{c}\bar{A}(\mathbf{u})\bar{b} \in [0,1]$ for all $\mathbf{u} = u_1u_2\dots u_{|\mathbf{u}|} \in \mathbb{Y}^*, |\mathbf{u}| \leq t$, where $\bar{A}(\mathbf{u}) := \bar{A}(u_1)\bar{A}(u_2)\dots\bar{A}(u_{|\mathbf{u}|})$. We use "pseudo Mealy HMM" instead of "t-pseudo Mealy HMM" whenever the horizon t is clear from the context. The matrix $\bar{A}_{\bar{\mathbb{Q}}}$ is called the *pseudo state transition matrix.* $\bar{c}$ is a vector in $\mathbb{R}^{1\times|\bar{\mathbb{Q}}|}$ called the *pseudo initial state distribution* for which $\bar{c}\bar{b} = 1$. The conditions $\bar{c}\bar{b} = 1$ and $\bar{A}_{\bar{\mathbb{Q}}}\bar{b} = \bar{b}$ are called *consistency conditions* of the pseudo HMM. The pseudo HMM is called *stationary* if the pseudo initial state distribution vector is a left eigenvector of the quasi state transition matrix corresponding to the eigenvalue 1: $\bar{c}\bar{A} = \bar{c}$.

The *pseudo string probabilities* $\bar{\mathcal{P}} : \mathbb{Y}^* \mapsto \mathbb{R}$ *generated by* a t-pseudo Mealy HMM $\left(\bar{\mathbb{Q}}, \mathbb{Y}, \bar{A}, \bar{c}, \bar{b}\right)$ are given by

$$\bar{\mathcal{P}}(\mathbf{u}) \quad = \quad \bar{c}\bar{A}(\mathbf{u})\bar{b},$$

where $\mathbf{u} = u_1u_2\dots u_{|\mathbf{u}|} \in \mathbb{Y}^*$ and where $\bar{A}(\mathbf{u}) := \bar{A}(u_1)\bar{A}(u_2)\dots\bar{A}(u_{|\mathbf{u}|})$. Notice that the string probabilities generated by a t-pseudo Mealy HMM take values in $[0,1]$ if the string length is smaller than or equal to t and in $\mathbb{R}$ if the string length is larger than t. We now formulate the partial pseudo Mealy realization problem and the partial minimal pseudo Mealy realization problem.

Problem 4.3 (partial pseudo Mealy realization problem)**.** *Given partial string probabilities* $\mathcal{P}^{(t)} : \{\mathbf{u} \in \mathbb{Y}^*, |\mathbf{u}| \leq t\} \mapsto [0,1]$*. Find a t-pseudo Mealy HMM* $(\bar{\mathbb{Q}}, \mathbb{Y}, \bar{A}, \bar{c}, \bar{b})$ *that generates* $\mathcal{P}^{(t)}$*.*

Problem 4.4 (minimal partial pseudo Mealy realization problem)**.** *Given partial string probabilities* $\mathcal{P}^{(t)} : \{\mathbf{u} \in \mathbb{Y}^*, |\mathbf{u}| \leq t\} \mapsto [0,1]$*. Find a t-pseudo Mealy HMM* $(\bar{\mathbb{Q}}, \mathbb{Y}, \bar{A}, \bar{c}, \bar{b})$*, with* $|\bar{\mathbb{Q}}|$ *as small as possible, that generates* $\mathcal{P}^{(t)}$*.*

As a consequence of Proposition 4.8, a partial pseudo realization always exists. However, it has not yet been shown that a minimal partial pseudo realization (a solution to Problem 4.4) always exists. Intuitively, since there always exist partial pseudo realizations, there exists at least one which has a smaller dimension than the others. We now prove that under some conditions, Algorithm 4.1 can be applied to solve the minimal partial pseudo realization problem.

Proposition 4.9. *The minimal partial pseudo realization problem of string probabilities* $\mathcal{P}^{(t)}$ *can be solved using Algorithm 4.1 if there exist integers* t_1 *and* t_2 *with* $t_1 + t_2 = t + 1$ *such that*

$$\operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2)} = \operatorname{rank} \mathfrak{H}_{(1:t_1+1,1:t_2)} = \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2+1)}. \tag{4.8}$$

Proof: Algorithm 4.1 can be employed with

$$\begin{aligned} r &= \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2)}, \\ M &= \mathfrak{H}_{(1:t_1,1:t_2)}, \\ R &= \mathfrak{H}_{(1,1:t_2)}, \\ K &= \mathfrak{H}_{(1:t_1,1)}. \end{aligned}$$

The resulting partial pseudo realization is clearly minimal. ■

Condition (4.8) is called the *rank condition.* We now ask the question whether two solutions to the minimal partial pseudo realization problem are connected by a similarity transformation (as in Proposition 3.2). Assume $\sigma = (\bar{\mathbb{Q}}, \mathbb{Y}, \bar{A}, \bar{c}, \bar{b})$ is a solution to the minimal partial pseudo realization problem of the partial probabilities $\mathcal{P}^{(t)}$. Any other solution is connected by a similarity transformation as in Proposition 3.2, if the pseudo extension of the string probabilities $\bar{\mathcal{P}}^{(\infty)}_{(t)}$ defined by $\bar{\mathcal{P}}^{(\infty)}_{(t)}(\mathbf{u}) = \bar{c}\bar{A}(\mathbf{u})\bar{b}, |\mathbf{u}| > t$, is the unique extension that yields a Hankel matrix of rank $|\bar{\mathbb{Q}}|$. In other words, if there exist two different pseudo extensions $\bar{\mathcal{P}}^{(\infty)}_{(t)}$ and $\bar{\mathcal{P}}'^{(\infty)}_{(t)}$, which give us the minimal partial realizations σ and σ' with Algorithm 4.1, then σ and σ' are not necessarily connected by a similarity transformation as in Proposition 3.2. We show that if the rank condition (4.8) holds, there exists only one pseudo extension of the string probabilities that yields a Hankel matrix with rank equal to $\operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2)}$.

Lemma 4.1. *Given matrices* $A \in \mathbb{R}^{a_1 \times a_2}$*,* $B \in \mathbb{R}^{a_1 \times b_2}$ *and* $C \in \mathbb{R}^{c_1 \times a_2}$ *such that*

$$\operatorname{rank} A = \operatorname{rank} \begin{bmatrix} A & B \end{bmatrix} = \operatorname{rank} \begin{bmatrix} A \\ C \end{bmatrix}.$$

If there exists a matrix $D \in \mathbb{R}^{c_1 \times b_2}$ for which

$$\operatorname{rank}\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \operatorname{rank}\begin{bmatrix} A & B \end{bmatrix} = \operatorname{rank}\begin{bmatrix} A \\ C \end{bmatrix} = \operatorname{rank} A,$$

then the matrix D is unique.

Proof: The proof of this lemma is given in [98]. ■

Using Lemma 4.1, we prove the following proposition.

Proposition 4.10. *Given partial string probabilities $\mathcal{P}^{(t)}$ satisfying*

$$\operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2)} = \operatorname{rank} \mathfrak{H}_{(1:t_1+1,1:t_2)} = \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2+1)},$$

for some t_1, t_2 with $t_1 + t_2 = t + 1$. Then the pseudo extension of the partial string probabilities $\bar{\mathcal{P}}_{(t)}^{(\infty)}$ to $\bar{\mathcal{P}}^{(t,\infty)}$, for which

$$\operatorname{rank} \mathfrak{H}_{(1:t_1+l,1:t_2+m)} = \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2)},$$

for $l = 0, 1, 2, \dots$ and $m = 0, 1, 2, \dots$, is unique.

Proof: By applying Lemma 4.1 with $A = \mathfrak{H}_{(1:t_1,1:t_2)}$, $B = \mathfrak{H}_{(1:t_1,t_2+1)}$, $C = \mathfrak{H}_{(t_1+1,1:t_2)}$, we conclude that there exists at most one matrix $D = \mathfrak{H}_{(t_1+1,t_2+1)}$ such that $\operatorname{rank} \mathfrak{H}_{(1:t_1+1,1:t_2+1)} = \operatorname{rank} \mathfrak{H}_{(1:t_1+1,1:t_2)} = \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2+1)} = \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2)}$.

Proceed by recursion. Suppose $\operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2+k)} = \operatorname{rank} \mathfrak{H}_{(1:t_1+1,1:t_2+k)} = \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2+k+1)} = \operatorname{rank} \mathfrak{H}_{(1:t_1+1,1:t_2+k+1)}$, for an integer k. It follows from the Hankel structure that $\operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2+k+1)} = \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2+k+2)}$, such that we have that $\operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2+k+1)} = \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2+k+2)} = \operatorname{rank} \mathfrak{H}_{(1:t_1+1,1:t_2+k+1)}$. Now from Lemma 4.1 with $A = \mathfrak{H}_{(1:t_1,1:t_2+k+1)}$, $B = \mathfrak{H}_{(1:t_1,t_2+k+2)}$, $C = \mathfrak{H}_{(t_1+1,1:t_2)}$, we conclude that there exists at most one matrix $D = \mathfrak{H}_{(t_1+1,t_2+k+2)}$ such that $\operatorname{rank} \mathfrak{H}_{(1:t_1+1,1:t_2+k+2)} = \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2+k+1)} = \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2+k+2)} = \operatorname{rank} \mathfrak{H}_{(1:t_1+1,1:t_2+k+1)}$. ■

We finally prove that any two minimal pseudo realizations of string probabilities $\mathcal{P}^{(t)}$ are connected by a similarity transformation.

Proposition 4.11. *If the rank condition (4.8) holds, then any two minimal partial pseudo realizations of string probabilities $\mathcal{P}^{(t)}$ are connected by a similarity transformation.*

Proof: Let $(\bar{\mathbb{Q}}, \mathbb{Y}, \bar{A}, \bar{c}, \bar{b})$ be the pseudo realization of $\mathcal{P}^{(t)}$ found using Algorithm 4.1. The pseudo extension of the string probabilities $\bar{\mathcal{P}}_{(t)}^{(\infty)}$ defined by $\bar{\mathcal{P}}_{(t)}^{(\infty)}(\mathbf{u}) = \bar{c}\bar{A}(\mathbf{u})\bar{b}$, $|\mathbf{u}| > t$ is such that

$$\operatorname{rank} \mathfrak{H}_{(1:t_1+l,1:t_2+m)} = \operatorname{rank} \mathfrak{H}_{(1:t_1,1:t_2)},$$

for $l = 0, 1, 2, \dots$ and $m = 0, 1, 2, \dots$. By Proposition 4.10, this extension is unique, which proves the proposition. ■

Up to now, we have solved the partial pseudo realization problem under the condition that the rank condition (4.8) holds. We now give some hints for the partial pseudo realization problem if the rank condition does not hold, however, we do not go into detail. In order to use Algorithm 4.1, a pseudo extension $\bar{\mathcal{P}}^{(T)}_{(t)}$ must be specified until $\operatorname{rank} \mathfrak{H}_{(1:T_1,1:T_2)} = \operatorname{rank} \mathfrak{H}_{(1:T_1+1,1:T_2)} = \operatorname{rank} \mathfrak{H}_{(1:T_1,1:T_2+1)}$, where $T_1 + T_2 = T + 1$. However, these matrices are partially arbitrary, which proves that, in case the rank condition does not hold, minimal partial pseudo realizations are not necessarily connected by a similarity transformation. In order to get an idea of the minimal dimension of a partial realization of the string probabilities $\mathcal{P}^{(t)}$, we consider the incomplete Hankel matrix $\mathfrak{H}^{(t)}_{(1:t,1:t)}$ associated with the partial string probabilities $\mathcal{P}^{(t)}$,

$$\mathfrak{H}^{(t)}_{(1:t,1:t)} = \begin{bmatrix} \mathfrak{H}_{(1,1)} & \mathfrak{H}_{(1,2)} & \mathfrak{H}_{(1,3)} & \cdots & \mathfrak{H}_{(1,t-3)} & \mathfrak{H}_{(1,t-2)} & \mathfrak{H}_{(1,t-1)} \\ \mathfrak{H}_{(2,1)} & \mathfrak{H}_{(2,2)} & \mathfrak{H}_{(2,3)} & \cdots & \mathfrak{H}_{(2,t-3)} & \mathfrak{H}_{(2,t-2)} & * \\ \mathfrak{H}_{(3,1)} & \mathfrak{H}_{(3,2)} & \mathfrak{H}_{(3,3)} & \cdots & \mathfrak{H}_{(3,t-3)} & * & * \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ \mathfrak{H}_{(t-3,1)} & \mathfrak{H}_{(t-3,2)} & \mathfrak{H}_{(t-3,3)} & \cdots & * & * & * \\ \mathfrak{H}_{(t-2,1)} & \mathfrak{H}_{(t-2,2)} & * & \cdots & * & * & * \\ \mathfrak{H}_{(t-1,1)} & * & * & \cdots & * & * & * \end{bmatrix}$$

where the positions indicated by $*$ are left blanc since no data is available. A lower bound for the dimension of a minimal partial realization of $\mathcal{P}^{(t)}$ can be obtained by counting the number of linear independent rows already in $\mathfrak{H}^{(t)}_{(1:t,1:t)}$. This number can only increase when the $*$'s are filled in.

4.3 Approximate partial pseudo realization

In the previous section we considered the partial realization problem where exact string probabilities of strings up to a certain length t are given. In practice however, usually only *approximate* string probabilities of strings *up to length t* are available (e.g. estimated from an output string using (3.3)). Because the string probabilities are not exact, the Hankel matrix generically has high rank which implies that it is not possible to find a low order pseudo HMM that exactly matches the string probabilities up to length t. Therefore, it may be more useful to make a good low order approximate realization of the string probabilities rather than to try to match them exactly. In this section we describe four different methods to solve the approximate partial pseudo realization problem.

The first three methods approximate the Hankel matrix $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(1:t_1,1:t_2)}$ of approximate string probabilities with a low rank Hankel matrix $(\mathfrak{H}_{\mathcal{P}})_{(1:t_1,1:t_2)}$.

In case the rank condition (4.8) holds for the low rank matrix $(\mathfrak{H}_{\mathcal{P}})_{(1:t_1,1:t_2)}$, Algorithm 4.1 can be applied to find an approximate partial pseudo realization. In case the rank condition does not hold, one needs to make first an extension of the string probabilities such that the rank condition holds, and subsequently apply Algorithm 4.1.

In Section 4.3.1 the rank of the Hankel matrix is reduced by projecting the approximate string probabilities on the consistency and/or stationarity constraints. In Section 4.3.2 the rank of the Hankel matrix is reduced by using optimal low rank matrix approximation techniques. In Section 4.3.3 the Hankel matrix $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(1:t_1,1:t_2)}$ is approximated by a low rank Hankel matrix of string probabilities that are both consistent and stationary. Clearly, this last method is a combination of the previous two methods.

In Section 4.3.4 finally, the fourth method is presented. The method aims at obtaining a full-order balanced realization of the approximate string probabilities and subsequently reduces the balanced realization to find an approximate pseudo realization of the approximate string probabilities.

4.3.1 Projecting on the consistency and/or stationarity constraints

The Hankel matrix $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(1:t_1,1:t_2)}$ with $t_1 + t_2 = t + 2$ of approximate string probabilities $\tilde{\mathcal{P}}^{(t)}$ typically has high rank. One of the reasons is that approximate string probabilities are not necessary consistent such that Proposition 4.1 does not hold. To reduce the rank of the Hankel matrix, we propose to project the approximate string probabilities $\tilde{\mathcal{P}}^{(t)}$ on the consistency constraints to find $\mathcal{P}^{(t)}$. The rank of the Hankel matrix associated with the consistent string probabilities $(\mathfrak{H}_{\mathcal{P}})_{(1:t_1,1:t_2)}$ is then smaller than or equal to the rank of $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(1:t_1,1:t_2)}$.

Projecting the string probabilities $\tilde{\mathcal{P}}^{(t)}$ on the consistency condition to obtain $\mathcal{P}^{(t)}$ can be done by rescaling the string probabilities of strings of length t such that $\sum_{\mathbf{u}\in\mathbb{Y}^t} \mathcal{P}^{(t)}(\mathbf{u}) = 1$ and then calculating the string probabilities of strings of length smaller than t from the string probabilities of strings of length t, using $\sum_{\mathbf{y}\in\mathbb{Y}} \mathcal{P}(\mathbf{uy}) = \mathcal{P}(\mathbf{u})$. Using Proposition 4.1, it is easily seen that the rank of the Hankel matrix $(\mathfrak{H}_{\mathcal{P}})_{(t_1,t_2)}$ is bounded above by $\min\{1 + |\mathbb{Y}|^1 + |\mathbb{Y}|^2 + \ldots + |\mathbb{Y}|^{t_1-1}, |\mathbb{Y}|^{t_2-1}\}$.

Another projection method to reduce the rank of the Hankel matrix, consists of projecting the string probabilities $\tilde{\mathcal{P}}^{(t)}$ on the stationarity and consistency constraints to obtain $\mathcal{P}^{(t)}$. It is clear from Proposition 4.2 that the rank of the Hankel matrix associated with the stationary and consistent string probabilities $(\mathfrak{H}_{\mathcal{P}})_{(1:t_1,1:t_2)}$ is smaller than or equal to the rank of $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(1:t_1,1:t_2)}$.

The projection on the stationarity and consistency constraints consists of two steps. In a first step the string probabilities of length t are projected on the stationarity and consistency constraints for strings of length t. In a second step, the string probabilities of strings of length smaller than t are calculated from the string probabilities of length t using $\sum_{\mathbf{y}\in\mathbb{Y}} \mathcal{P}(\mathbf{uy}) = \mathcal{P}(\mathbf{u})$. From Proposition 4.2, the rank of the Hankel matrix associated with the consistent string probabilities $(\mathfrak{H}_{\mathcal{P}})_{(1:t_1,1:t_2)}$ is bounded above by $\min\{|\mathbb{Y}|^{t_1-1}, |\mathbb{Y}|^{t_2-1}\}$.

We now explain how projection on the stationarity and consistency constraints can be carried out. Stationary and consistent string probabilities of length t obey the following constraints

$$\begin{aligned} \sum_{\mathbf{u}\in\mathbb{Y}^{t-|\mathbf{v}|}} \mathcal{P}(\mathbf{uv}) &= \sum_{\mathbf{u}\in\mathbb{Y}^{t-|\mathbf{v}|}} \mathcal{P}(\mathbf{vu}), \quad \forall\, \mathbf{v}\in\mathbb{Y}^*, |\mathbf{v}|\le t, && (4.9)\\ \sum_{\mathbf{u}\in\mathbb{Y}^t} \mathcal{P}(\mathbf{u}) &= 1. && (4.10)\end{aligned}$$

For $t=3$ and $\mathbb{Y}=\{0,1\}$, the constraints (4.9) can be written explicitly as (in front of each equation, we give the values of $\mathbf{v}$)

$$\begin{array}{c} 0\\ 1\\ \hline 00\\ 01\\ 10\\ 11 \end{array} \quad \left[\begin{array}{cccccccc} 0 & 1 & 0 & 1 & -1 & 0 & -1 & 0\\ 0 & -1 & 0 & -1 & 1 & 0 & 1 & 0\\ \hline 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0\\ 0 & -1 & 1 & 1 & 0 & -1 & 0 & 0\\ 0 & 0 & -1 & 0 & 1 & 1 & -1 & 0\\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \end{array}\right] \left[\begin{array}{c} \mathcal{P}(000)\\ \mathcal{P}(001)\\ \mathcal{P}(010)\\ \mathcal{P}(011)\\ \mathcal{P}(100)\\ \mathcal{P}(101)\\ \mathcal{P}(110)\\ \mathcal{P}(111) \end{array}\right] = 0.$$

For general t, the constraint (4.9) becomes

$$S\mathfrak{p}_{(t)} = 0,$$

where $\mathfrak{p}_{(t)} = \mathfrak{H}^\top_{(1,t+1)}$ denotes the vector of lexicographically ordered string probabilities of strings of length t and

$$\begin{aligned} S &= \begin{bmatrix} S(1)\\ S(2)\\ \vdots\\ S(t-2) \end{bmatrix},\\ S(i) &= I_{|\mathbb{Y}|^i}\otimes \mathbf{1}_{1\times|\mathbb{Y}|^{t-i}} - \mathbf{1}_{1\times|\mathbb{Y}|^{t-i}}\otimes I_{|\mathbb{Y}|^i}. \end{aligned}$$

The problem is to find a vector of consistent and stationary string probabilities $\mathfrak{p}_{(t)}$ that is close to the given vector of approximate string probabilities $\tilde{\mathfrak{p}}_{(t)}$, i.e. to solve

$$\begin{aligned} \mathfrak{p}_{(t)} = \quad & \operatorname{argmin}_{\dot{\mathfrak{p}}_{(t)}} \quad D(\tilde{\mathfrak{p}}_{(t)}, \dot{\mathfrak{p}}_{(t)})\\ & \text{such that} \quad S\dot{\mathfrak{p}}_{(t)} = 0, \end{aligned}$$

or, equivalently

$$\mathfrak{p}_{(t)} = \quad \ker S \operatorname{argmin}_{\dot{x}} \quad D(\tilde{\mathfrak{p}_{(t)}}, \ker S\dot{x}),$$

where $\ker(X)$ denotes the kernel of the matrix X. By taking $D(X,Y)$ to be the square of the Frobenius distance between X and Y, i.e. $D(X,Y) = ||X-Y||_F^2$, the problem becomes a least squares problem, such that the solution is given by $\mathfrak{p}_{(t)} = \ker S (\ker S)^\dagger \tilde{\mathfrak{p}}_{(t)}$.

Up to now, we described a method to project approximate string probabilities of strings of length t on the stationarity constraints. To make sure that the consistency constraints hold, it suffices to rescale them such that $\sum_{\mathbf{u}\in\mathbb{Y}^t} \mathcal{P}(\mathbf{u}) = 1$. The string probabilities of length smaller than t can be calculated from the string probabilities of length t as before. In this way the string probabilities $\mathcal{P}^{(t)}$ are consistent and stationary, and can be stacked in the Hankel matrix $(\mathfrak{H}_\mathcal{P})_{(1:t_1,1:t_2)}$.

4.3.2 Low rank approximation of largest Hankel block

Another approach to build a low rank Hankel matrix that is close to a Hankel matrix of approximate string probabilities $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(1:t_1,1:t_2)}$, is by computing $(\mathfrak{H}_\mathcal{P})_{(t_1,t_2)}$ as a rank a approximation of the largest block $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(t_1,t_2)}$ of the Hankel matrix $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(1:t_1,1:t_2)}$. This is equivalent to taking $(\mathfrak{H}_\mathcal{P})_{(t_1,t_2)} = VH$ where V and H are computed from

$$(V,H) = \operatorname{argmin}_{\dot{V},\dot{H}} \quad D((\mathfrak{H}_{\tilde{\mathcal{P}}})_{(t_1,t_2)}, \dot{V}\dot{H}), \tag{4.11}$$

where the inner dimension of the approximation $\dot{V}\dot{H}$ is equal to a. In a second step, all other string probabilities of $(\mathfrak{H}_\mathcal{P})_{(1:t_1,1:t_2)}$ are calculated from the string probabilities in $(\mathfrak{H}_\mathcal{P})_{(t_1,t_2)}$. The rank of the resulting Hankel matrix is bounded above by $\min\{1 + |\mathbb{Y}|^1 + |\mathbb{Y}|^2 + \ldots + |\mathbb{Y}|^{t_1-1} + a, |\mathbb{Y}|^{t_2}\}$.

For the low rank approximation of the largest Hankel block, the nonnegative matrix factorization without nonnegativity constraints on the factors (see Section 2.5) is used. This method approximates a nonnegative matrix with a low rank product VH in an optimal way (with respect to the Kullback-Leibler divergence), subject to the constraint that VH is elementwise nonnegative, but without nonnegativity constraints on V and H separetely. This method is well-suited for the problem at hand as it guarantees that the approximate string probabilities are nonnegative. In addition, from Property 2.1 it follows that a consistent Hankel block, i.e. a Hankel block with element sum equal to 1, is approximated by a low rank consistent Hankel block.

4.3.3 Structured low rank approximation of largest Hankel block

So far we described two methods for obtaining a low rank approximation $(\mathfrak{H}_\mathcal{P})_{(1:t_1,1:t_2)}$ of the Hankel matrix $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(1:t_1,1:t_2)}$. Both methods first approximate the largest Hankel block and subsequently compute the complete Hankel matrix from the largest block. The first method makes sure that the string probabilities in the largest Hankel block are consistent and stationary. In this way, the rank of the Hankel matrix is equal to the rank of the largest Hankel block. This gives a rank reduction, but it is possible that the rank is still high. The second method makes a low rank approximation of the largest Hankel block where the rank can be chosen by the user. However, because this approximation is not necessary stationary, the rank of the total Hankel matrix can be larger than the rank of the largest block. In this section we propose a method that combines the advantages of both methods by approximating the largest block of string proabilities of the Hankel matrix with a rank a matrix that contains stationary string probabilities. By doing so the rank of the total Hankel matrix is equal to a, and is completely user-defined.

The matrix approximation problem for the Hankel block $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(t_1,t_2)}$, can be written as

$$\begin{array}{ll} \min_{\dot{V},\dot{H}} & D((\mathfrak{H}_{\tilde{\mathcal{P}}})_{(t_1,t_2)}, \dot{V}\dot{H}), \\ \text{such that} & S\dot{\mathfrak{p}}_t = 0, \end{array} \tag{4.12}$$

where the inner dimension a of the approximation $\dot{V}\dot{H}$ can be chosen by the user and where $\dot{\mathfrak{p}}_t$ is the vector of lexicographically ordered approximate length-t string probabilities stacked in the largest Hankel block $\dot{V}\dot{H}$. We use the method of [26] to solve this problem in the Frobenius distance. That method approximates a matrix P with a low rank matrix that obeys linear structural constraints. The method supposes that the set of all rank a matrices forms a surface $\mathcal{R}(a)$ and the set Ω comprising matrices with the specified structure forms another surface (see Figure 4.2). Then any point on the intersection of these two geometric entities, forms a solution to problem (4.12). The approach of [26] is a linearly convergent method that finds points of this intersection. The approach is to alternate projections on the set of rank a matrices and on the set of matrices that fulfill the structural constraints while reducing the distance in between both kind of projections. Below, we decribe the algorithm of [26] (see Figure 4.2).

Algorithm 4.1. *Start with $A^{(0)} = P$, iterate the following two steps for $t = 0, 1, ...$ until convergence:*

1. *Calculate the rank a matrix $B^{(t)}$ in $\mathcal{R}(k)$ that is nearest to $A^{(t)}$ in Frobenius distance.*
2. *Compute the projection $A^{(t+1)}$ of $B^{(t)}$ onto the subspace Ω.*

It is clear that in the algorithm described above, the matrices $A^{(0)}, A^{(1)}, A^{(2)}, ...$ will not necessarily have the desired rank a. However, it is clear that

$$||A^{(t+1)} - B^{(t+1)}|| \leq ||A^{(t+1)} - B^{(t)}|| \leq ||A^{(t)} - B^{(t)}||.$$

So Algorithm 4.1 is a descent method. In our problem the first step of the algorithm can be carried out by use of the truncated singular value decomposition and the second step is a least squares problem as explained in Section 4.3.1. In case the element sum of the obtained structured low rank approximation differs from 1, a rescaling can be performed to overcome this problem. As a result, we have a method to approximate the largest Hankel block $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(t_1,t_2)}$ with a rank a block $(\mathfrak{H}_{\mathcal{P}})_{(t_1,t_2)}$ of consistent and stationary string probabilities. The complete Hankel matrix, built from $(\mathfrak{H}_{\mathcal{P}})_{(t_1,t_2)}$ as before, is of rank a.

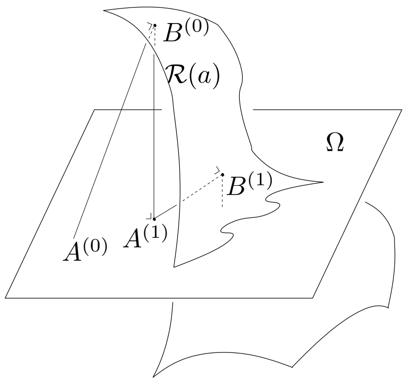


Figure 4.2: *Visualisation of $\mathcal{R}(a)$, the set of all rank a matrices and Ω, the set comprising matrices with a specified structure. Algorithm 4.1 converges to a point of the intersection of $\mathcal{R}(k)$ and Ω.*

4.3.4 Balanced approximate quasi realization

In this section we present a new exact quasi realization algorithm (Algorithm 4.2) which is a special case of Algorithm 4.1. We show that a quasi realization obtained with Algorithm 4.2 has an interesting property: it is balanced (see further). A balanced quasi realization can be easily reduced to a lower order quasi realization. The exact quasi realization algorithm followed by a model reduction step, gives rise to another approximate realization algorithm.

We start with Algorithm 4.2, which is a special case of Algorithm 4.1.

Algorithm 4.2. *Consider a rank r Hankel matrix $\mathfrak{H}$ ($r < \infty$) of string probabilities $\mathcal{P}$ of a process taking values in the finite set $\mathbb{Y}$. Perform the following steps.*

1. *For each $y \in \mathbb{Y}$ define $\sigma_y\mathfrak{H}$ as $\mathfrak{H}_{(\sigma_y\mathcal{U},\mathcal{V})}$, where $\sigma_y\mathcal{U}$ is the set of all strings $\mathbf{u}y$, $\mathbf{u} \in \mathbb{Y}^*$ in first lexicographical ordering and where $\mathcal{V}$ is the set of all strings of $\mathbb{Y}^*$ in last lexicographical ordering. Equivalently, by the Hankel structure, $\sigma_y\mathfrak{H}$ can be defined as $\mathfrak{H}_{(\mathcal{U},\sigma_y\mathcal{V})}$, where $\mathcal{U}$ is the set of all strings of $\mathbb{Y}^*$ in first lexicographical ordering and where $\sigma_y\mathcal{V}$ is the set of all strings $y\mathbf{u}$, $\mathbf{u} \in \mathbb{Y}^*$ in last lexicographical ordering.*
2. *Compute the reduced SVD $\mathfrak{H} = U\Sigma V^\top = U\sqrt{\Sigma}\sqrt{\Sigma}V^\top$, with $\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$.*

3. A minimal quasi realization $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ is now obtained as follows:

$$\begin{aligned}
\mathbb{Q} &= \{1, 2, \dots, r\}, \\
A(y) &= \sqrt{\Sigma^{-1}} U^\top \sigma_y \mathfrak{H} V \sqrt{\Sigma^{-1}}, \quad \forall\, y \in \mathbb{Y}, \\
c &= \mathfrak{H}_{(1,1:\infty)} V \sqrt{\Sigma^{-1}}, \\
b &= \sqrt{\Sigma^{-1}} U^\top \mathfrak{H}_{(1:\infty,1)}.
\end{aligned}$$

We now define the notion of balanced quasi realizations. Therefore, for a quasi HMM $(\mathbb{Q}, \mathbb{Y}, A, c, b)$, define the matrices $W(A, b)$ and $M(c, A)$ as:

$$\begin{aligned}
W(A, b) &:= \sum_{\mathrm{y} \in \mathbb{Y}^*} A(\mathrm{y}) b b^\top A(\mathrm{y})^\top = \mathcal{C}(A, b)\mathcal{C}(A, b)^\top, \\
M(c, A) &:= \sum_{\mathrm{y} \in \mathbb{Y}^*} A(\mathrm{y})^\top c^\top c A(\mathrm{y}) = \mathcal{O}(c, A)^\top \mathcal{O}(c, A).
\end{aligned}$$

Note that $W(A, b)$ and $M(c, A)$ are the analogues of the controllability and observability Gramians of linear time-invariant systems. Obviously, $W(A, b) = W(A, b)^\top \geq 0$ and $M(c, A) = M(c, A)^\top \geq 0$. Moreover, if $\mathcal{C}(A, b)$ has full row rank and $\mathcal{O}(c, A)$ has full column rank (as is the case for minimal quasi-realizations), then the strict inequality holds.

We assume that the infinite sums in the definitions above, are finite. It is an intersting point of further research to check under which conditions on $A(\mathrm{y}), \mathrm{y} \in \mathbb{Y}$ this assumption is fulfilled.

If the matrices $W(A, b)$ and $M(c, A)$ are finite, then it is easy to verify that they are solutions to the Lyapunov equations:

$$\sum_{\mathrm{y} \in \mathbb{Y}} A(\mathrm{y}) W(A, b) A(\mathrm{y})^\top - W(A, b) = -b b^\top, \tag{4.13}$$

$$\sum_{\mathrm{y} \in \mathbb{Y}} A(\mathrm{y})^\top M(c, A) A(\mathrm{y}) - M(c, A) = -c^\top c. \tag{4.14}$$

Now, a quasi hidden Markov model is called *balanced* if the matrices $W(A, b)$ and $M(c, A)$ are diagonal and equal to each other. It can be shown that for every quasi realization, there exists an equivalent balanced quasi realization. We now first show that the quasi realization obtained with Algorithm 4.2 is balanced. Subsequently, we show that a balanced quasi realization of order r can be reduced to a quasi realization of order s with $s < r$.

Proposition 4.12. *The quasi realization $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ obtained using Algorithm 4.2 is balanced.*

Proof: We first prove that

$$\begin{aligned}
\mathcal{O}(c, A)_{1,:} &= \mathfrak{H}_{(1,1:\infty)} V \sqrt{\Sigma^{-1}}, \\
&= (\mathfrak{H}_{(1:\infty,1:\infty)} V \sqrt{\Sigma^{-1}})_{1,:},
\end{aligned}$$

$$= (U\sqrt{\Sigma})_{1,:},$$

and

$$\begin{aligned}
\mathcal{O}(c,A)_{2,:} &= \mathfrak{H}_{(1,1:\infty)} V\sqrt{\Sigma^{-1}}\sqrt{\Sigma^{-1}}U^\top \sigma_{\mathrm{y}_1}\mathfrak{H}V\sqrt{\Sigma^{-1}},\\
&= (\sigma_{\mathrm{y}_1}\mathfrak{H}V\sqrt{\Sigma^{-1}})_{1,:},\\
&= (\mathfrak{H}V\sqrt{\Sigma^{-1}})_{2,:},\\
&= (U\sqrt{\Sigma})_{2,:}.
\end{aligned}$$

By continuing in this way, it follows that $\mathcal{O}(c,A) = U\sqrt{\Sigma}$. Analogously, it can be proven that $\mathcal{C}(A,b) = \sqrt{\Sigma}V^\top$. It follows that

$$\begin{aligned}
W(A,b) = \mathcal{C}(A,b)\mathcal{C}(A,b)^\top = \sqrt{\Sigma}V^\top\left(\sqrt{\Sigma}V^\top\right)^\top &= \Sigma,\\
M(c,A) = \mathcal{O}(c,A)^\top\mathcal{O}(c,A) = \left(U\sqrt{\Sigma}\right)^\top U\sqrt{\Sigma} &= \Sigma,
\end{aligned}$$

which proves the proposition. ■

A balanced reduced realization of order $s < r$ is now given by $(\mathbb{Q},\mathbb{Y},A,c,b)$ where

$$\begin{aligned}
\mathbb{Q} &= \{1,2,\ldots,s\},\\
A(\mathrm{y}) &= \sqrt{\Sigma_{(1)}^{-1}}U_{(1)}^\top\sigma_{\mathrm{y}}\mathfrak{H}V_{(1)}\sqrt{\Sigma_{(1)}^{-1}}, \quad \forall\, \mathrm{y}\in\mathbb{Y},\\
c &= \mathfrak{H}_{(1,1:\infty)}V_{(1)}\sqrt{\Sigma_{(1)}^{-1}},\\
b &= \sqrt{\Sigma_{(1)}^{-1}}U_{(1)}^\top\mathfrak{H}_{(1:\infty,1)}.
\end{aligned} \tag{4.15}$$

where the reduced SVD of $\mathfrak{H}$ is given by

$$\mathfrak{H} = \begin{bmatrix} U_{(1)} & U_{(2)} \end{bmatrix}\begin{bmatrix} \Sigma_{(1)} & 0\\ 0 & \Sigma_{(2)}\end{bmatrix}\begin{bmatrix} V_{(1)} & V_{(2)}\end{bmatrix}^\top, \tag{4.16}$$

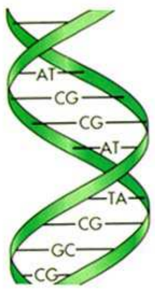
with $\Sigma_{(1)} = \operatorname{diag}(\sigma_1,\sigma_2,\ldots,\sigma_s)$ and $\Sigma_{(2)} = \operatorname{diag}(\sigma_{s+1},\sigma_{s+2},\ldots,\sigma_r)$, with $\sigma_1 \geq \sigma_2 \geq \ldots\sigma_s \gg \sigma_{s+1} \geq \ldots\sigma_r > 0$.

If only string probabilities of strings up to length t are given (the approximate partial realization problem), then Algorithm 4.2 that uses $\mathfrak{H}_{(1:t_1,1:t_2)}$ with $t_1 + t_2 = t+2$ instead of $\mathfrak{H}$ yields a realization that is almost balanced. We say that a realization is *almost balanced* if the diagonal elements of $W(A,b)$ and $M(c,A)$ are large compared to the off-diagonal elements and in addition $W(A,b)$ is close to $M(c,A)$. As a heuristic method for approximate partial realization, formulas (4.15) can be used.

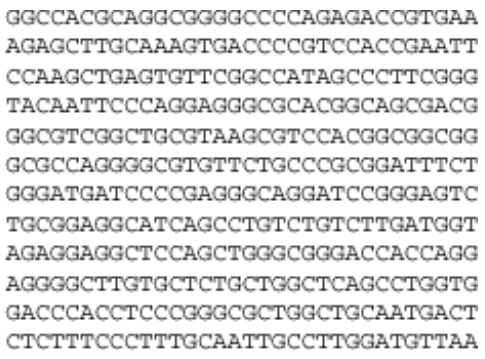
4.4 Modeling DNA sequences

In this section we apply the approximate partial quasi realization algorithms to the modeling of DNA sequences.

Desoxyribonucleic acid (DNA) is a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms. DNA forms a double helix of two anti-parallel chains with complementary nucleotide sequences. In Figure 4.3(a), the double DNA helix is schematically shown. The building blocks of the nucleotide sequences are the following four nucleotides: adenine (A), cytosine (C), guanine (G) and thymine (T). The human genome consists of approximately 3 billion nucleotide pairs. In Figure 4.3(b), an example of a part of a DNA sequence is shown. In some applications, it is important to have a quasi or positive hidden Markov model of a DNA sequence (for instance motif detection, explained in Section 7.7).



(a)



```
GGCCACGCAGGCGGGGCCCCAGAGACCGTGAA
AGAGCTTGCAAAGTGACCCCGTCCACCGAATT
CCAAGCTGAGTGTTCGGCCATAGCCCTTCGGG
TACAATTCCCAGGAGGGCGCACGGCAGCGACG
GGCGTCGGCTGCGTAAGCGTCCACGGCGGCGG
GCGCCAGGGGCGTGTTCTGCCCGCGGATTTCT
GGGATGATCCCCGAGGGCAGGATCCGGGAGTC
TGCGGAGGCATCAGCCTGTCTGTCTTGATGGT
AGAGGAGGCTCCAGCTGGGCGGGACCACCAGG
AGGGGCTTGTGCTCTGCTGGCTCAGCCTGGTG
GACCCACCTCCCGGGCGCTGGCTGCAATGACT
CTCTTTCCCTTTGCAATTGCCTTGGATGTTAA
```

(b)

Figure 4.3: *Desoxyribonucleic acid (DNA) is a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms. DNA forms a double helix of two anti-parallel chains with complementary nucleotide sequences (Subfigure (a)) . The human genome consists of approximately 3 billion nucleotide pairs. In Subfigure (b), an example of a DNA sequence of length 384 is shown.*

In this example we build a quasi hidden Markov model of the 40 sequences $\mathbf{u}_1, \dots \mathbf{u}_{40}$ of length 200 from `http://www.stat.ucla.edu/∼zhou/CisModule/` [119]. These sequences are the background sequences used in Section 7.7. The goal of this section is to make one quasi model of the 40 sequences. So the 40 sequences have to be considered as 40 different output sequences from one unknown (quasi) hidden Markov model.

For making a quasi HMM representation of the sequences, we calculate an estimate of the string probabilities $\tilde{\mathcal{P}}_{\mathbf{u}_i}$ of string up to length 4 for $\mathbf{u}_i$, $i = 1, 2, \dots, 40$ by using Equation 3.3. Subsequently, the mean string probabilities are calculated as $\tilde{\mathcal{P}} = \sum_{i=1}^{40} \tilde{\mathcal{P}}_{\mathbf{u}_i}$. In Table 4.1, we show the string probabilities of a selection of the strings of length 4 (the strings starting with A or C). The string probabilities of length smaller than 4 are computed from the string probabilities of length 4.

Table 4.1: *The approximate quasi realization algorithm is used to model* 40 *DNA sequences with symbols from* $\mathbb{Y} = \{A, C, G, T\}$. *Therefore the string probabilities of strings up to length 4 are calculated. We here show the string probabilities of a selection of the* 256 *strings of length 4 (the strings starting with A or C). The number at position* i, j *in the table is the probability of the string formed by concatenating the string that indexes row* i *with the symbol that indexes row* j.

	A	C	G	T
AAA	0.0045	0.0031	0.0051	0.0027
CAA	0.0035	0.0030	0.0050	0.0024
ACA	0.0030	0.0028	0.0043	0.0020
CCA	0.0040	0.0059	0.0066	0.0041
AGA	0.0065	0.0035	0.0065	0.0018
CGA	0.0013	0.0016	0.0036	0.0011
ATA	0.0020	0.0007	0.0015	0.0017
CTA	0.0022	0.0017	0.0013	0.0011
AAC	0.0048	0.0050	0.0061	0.0044
CAC	0.0053	0.0074	0.0086	0.0026
ACC	0.0007	0.0023	0.0020	0.0016
CCC	0.0037	0.0072	0.0083	0.0020
AGC	0.0060	0.0070	0.0089	0.0050
CGC	0.0049	0.0076	0.0069	0.0023
ATC	0.0027	0.0032	0.0028	0.0020
CTC	0.0039	0.0085	0.0082	0.0048
AAG	0.0033	0.0022	0.0010	0.0026
CAG	0.0039	0.0062	0.0029	0.0028
ACG	0.0034	0.0038	0.0026	0.0034
CCG	0.0075	0.0126	0.0071	0.0078
AGG	0.0049	0.0064	0.0051	0.0035
CGG	0.0034	0.0065	0.0053	0.0028
ATG	0.0025	0.0032	0.0006	0.0024
CTG	0.0043	0.0088	0.0020	0.0051
AAT	0.0027	0.0024	0.0021	0.0017
CAT	0.0014	0.0031	0.0039	0.0033
ACT	0.0013	0.0020	0.0023	0.0035
CCT	0.0017	0.0088	0.0094	0.0038
AGT	0.0015	0.0035	0.0030	0.0042
CGT	0.0009	0.0033	0.0026	0.0008
ATT	0.0010	0.0022	0.0031	0.0040
CTT	0.0013	0.0053	0.0034	0.0048

Now the string probabilities of strings up to length 4 are stacked in the

Hankel matrix $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(1:3,1:3)}$. The Hankel matrix has dimensions 21×21. The singular values of $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(1:3,1:3)}$ are plotted in Figure 4.4. The first four singular values are more dominant compared to the others. For that reason it can be expected that a fourth order model gives a good trade-off between accuracy and model complexity.

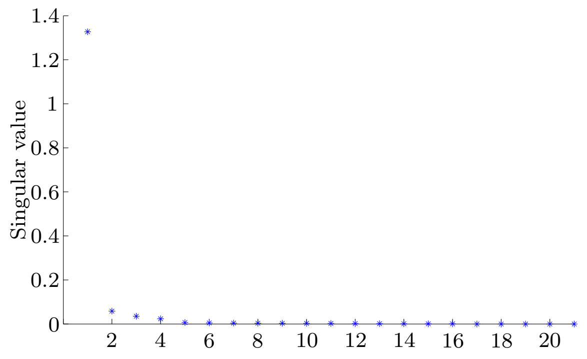


Figure 4.4: *Plot of the singular values of* $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(1:3,1:3)}$. *The first 4 singular values are dominant as compared to the others. For that reason it can be expected that a quasi hidden Markov model of order 4 gives a good trade-off between accuracy and complexity.*

Using the approximate quasi realization algorithm of Section 4.3.3, we build a quasi Mealy model of order 1 to 7. The Kullback-Leibler divergence between the Hankel block $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(3,3)}$ and the corresponding block of the Hankel matrix generated by the obtained model is given in Table 4.2. It follows that the fourth order model gives the best trade-off between accuracy and complexity.

Table 4.2: *Kullback-Leibler divergence between the Hankel block* $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(3,3)}$ *and the corresponding block of the Hankel matrix generated by the obtained quasi Mealy hidden Markov model of order* 1 *to* 7*.*

order	1	2	3	4	5	6	7
KL-divergence	0.1109	0.0653	0.0449	0.0263	0.0220	0.0211	0.0210

To get some intuition about the size of the Kullback-Leibler divergences, we plot a histogram of the Kullback-Leibler divergence between $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(3,3)}$ and 10000 random matrices with element sum equal to 1 (Figure 4.5). It is clear that the values for the Kullback-Leibler divergence in Table 4.2 are much smaller than the values of Figure 4.5. This is a first indication that the quasi modeling gives good results.

The obtained fourth order quasi model is given by $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ with

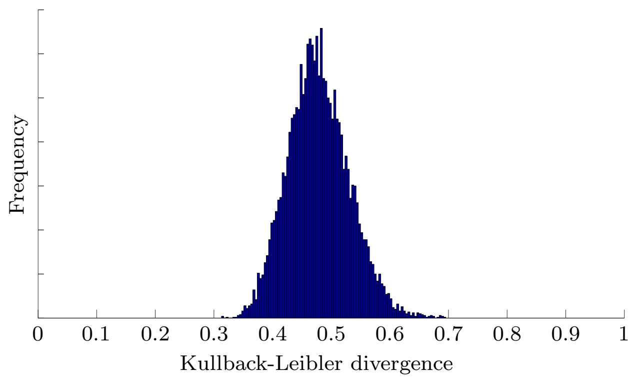


Figure 4.5: *We plot a histogram of the Kullback-Leibler divergence between* $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(3,3)}$ *and* 10000 *random matrices with element sum equal to* 1*. It is clear that the values for the Kullback-Leibler divergence in Table 4.2 are much smaller than the values in this histogram. This is a first indication that the quasi modeling gives good results.*

$$
\begin{aligned}
A(A) &= \begin{bmatrix} 0.2052 & 0.0213 & 0.0053 & -0.0064 \\ 0.0183 & 0.0794 & 0.0560 & -0.0641 \\ 0.8611 & 0.3508 & -0.0258 & 0.1431 \\ -0.4867 & 0.1467 & -0.0397 & 0.2249 \end{bmatrix}, \\
A(C) &= \begin{bmatrix} 0.2896 & -0.0314 & 0.0052 & -0.0014 \\ -0.5948 & 0.0711 & 0.0523 & -0.0512 \\ 0.1144 & 0.3067 & 0.0088 & 0.0977 \\ 0.7779 & 0.1010 & -0.0118 & 0.3361 \end{bmatrix}, \\
A(G) &= \begin{bmatrix} 0.3014 & 0.0110 & 0.0062 & 0.0082 \\ 0.8803 & 0.0432 & 0.0740 & -0.0982 \\ -0.2215 & 0.0522 & 0.1024 & 0.0639 \\ 0.2938 & -0.0543 & 0.1503 & 0.2376 \end{bmatrix}, \\
A(T) &= \begin{bmatrix} 0.2039 & 0.0012 & -0.0176 & 0.0005 \\ -0.3538 & -0.0564 & 0.0657 & -0.0878 \\ -0.6877 & 0.2149 & 0.1001 & 0.1110 \\ -0.5732 & 0.1030 & 0.0670 & 0.2628 \end{bmatrix}, \\
c &= \begin{bmatrix} -1.1218 & -0.0015 & 0.0010 & 0.0006 \end{bmatrix}, \\
b &= \begin{bmatrix} -0.8914 & 0.0255 & -0.0090 & 0.0677 \end{bmatrix}^\top.
\end{aligned}
$$

To check the quality of the quasi model, we compute the string probabilities generated by $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ for strings of length 4. A selection of these string probabilities (strings starting with A or C) is shown in Table 4.3. By comparing these probabilities to the probabilities of Table 4.1, we conclude that the modeling works well.

Table 4.3: *We show the string probabilities (for strings starting with A or C) generated by the quasi hidden Markov model* $(\mathbb{Q}, \mathbb{Y}, A, c, b)$ *of order 4 that is obtained by the quasi realization algorithm. By comparing these probabilities to the original string probabilities in Table 4.1, we conclude that the modeling works well.*

	A	C	G	T
AAA	0.0043	0.0022	0.0057	0.0023
CAA	0.0035	0.0023	0.0051	0.0019
ACA	0.0029	0.0027	0.0044	0.0023
CCA	0.0038	0.0052	0.0075	0.0034
AGA	0.0050	0.0035	0.0076	0.0029
CGA	0.0016	0.0019	0.0046	0.0007
ATA	0.0023	0.0014	0.0011	0.0019
CTA	0.0022	0.0015	0.0014	0.0016
AAC	0.0030	0.0020	0.0014	0.0022
CAC	0.0039	0.0055	0.0031	0.0036
ACC	0.0030	0.0041	0.0024	0.0034
CCC	0.0069	0.0135	0.0077	0.0082
AGC	0.0048	0.0060	0.0038	0.0044
CGC	0.0033	0.0068	0.0047	0.0029
ATC	0.0023	0.0027	0.0000	0.0029
CTC	0.0043	0.0079	0.0025	0.0049
AAG	0.0052	0.0045	0.0067	0.0039
CAG	0.0057	0.0066	0.0079	0.0039
ACG	0.0014	0.0018	0.0031	0.0015
CCG	0.0036	0.0073	0.0079	0.0028
AGG	0.0060	0.0066	0.0087	0.0044
CGG	0.0048	0.0070	0.0068	0.0024
ATG	0.0019	0.0019	0.0032	0.0025
CTG	0.0055	0.0076	0.0079	0.0044
AAT	0.0019	0.0014	0.0020	0.0026
CAT	0.0017	0.0031	0.0032	0.0030
ACT	0.0014	0.0025	0.0037	0.0028
CCT	0.0021	0.0072	0.0087	0.0054
AGT	0.0023	0.0034	0.0044	0.0038
CGT	0.0006	0.0029	0.0031	0.0011
ATT	0.0018	0.0026	0.0023	0.0037
CTT	0.0020	0.0051	0.0038	0.0047

4.5 The linear stochastic case

In this section we present a solution to the minimal linear stochastic realization problem (Problem 3.8). We only consider the exact linear stochastic realization problem, not the partial nor the partial approximate linear stochastic realization problem. We first present an answer to the realizability question. Subsequently, we describe how to find the minimal order of a linear stochastic realization and finally, we present a linear stochastic realization algorithm. It will become clear that the hidden Markov realization theory is very similar to the linear stochastic realization theory.

Define the *Hankel matrix* $\mathfrak{H}_\Lambda$ of autocovariances Λ as a doubly infinite matrix with

$$(\mathfrak{H}_\Lambda)_{ij} := \Lambda(i+j-1).$$

It is clear that the Hankel matrix looks like

$$\mathfrak{H}_\Lambda = \begin{bmatrix} \Lambda(1) & \Lambda(2) & \Lambda(3) & \dots \\ \Lambda(2) & \Lambda(3) & \Lambda(4) & \dots \\ \Lambda(3) & \Lambda(4) & \Lambda(5) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

The linear stochastic realizability question is solved by the following theorem.

Theorem 4.3. *An autocovariance sequence Λ is realizable by a linear stochastic model (A, C, P, Q, R, S) where $A \in \mathbb{R}^{n\times n}$ with $n < \infty$, if and only if the rank of the Hankel matrix $\mathfrak{H}_\Lambda$ is finite.*

Note that Theorem 4.3 is the equivalent of Theorem 4.1 for hidden Markov models. The following theorem provides a way to find the minimal order of a linear stochastic realization of a given autocovariance sequence.

Theorem 4.4. *The order of a minimal linear stochastic realization of Λ is equal to the rank of the Hankel matrix $\mathfrak{H}_\Lambda$.*

Theorem 4.4 is the linear stochastic equivalent of Theorem 4.2 for hidden Markov models. We now present a realization algorithm for linear stochastic models. It turns out that the positive definiteness of the matrices P, Q and R comes for free in the realization algorithm. As explained before, this is the reason why we do not make a distinction between quasi linear stochastic realization and positive linear stochastic realization as we did for hidden Markov models.

Algorithm 4.3. *Given a rank r Hankel matrix $\mathfrak{H}_\Lambda$ ($r < \infty$) of autocovariances Λ of a process over $\mathbb{Z}_+$ taking values in $\mathbb{R}^p$. Perform the following steps.*

1. *Find a submatrix M of $\mathfrak{H}_\Lambda$ with $\operatorname{rank} M = \operatorname{rank} \mathfrak{H}_\Lambda$. Suppose M is formed by the elements in the rows $r_1, r_2, \dots, r_m$ and columns $c_1, c_2, \dots, c_n$.*
2. *Define R as the submatrix of $\mathfrak{H}_\Lambda$ formed by the elements in the first p rows and columns $c_1, c_2, \dots, c_n$ and K as the submatrix of $\mathfrak{H}_\Lambda$ formed by the*

elements in the first p columns and rows $r_1, r_2, \ldots, r_m$. Finally, define σM as the submatrix of $\mathfrak{H}_\Lambda$ with elements of rows $r_1, r_2, \ldots, r_m$ and columns $c_1+p, c_2+p, \ldots, c_n+p$. Equivalently, by the Hankel structure, σM can be defined as the submatrix of $\mathfrak{H}_\Lambda$ with elements of rows $r_1+p, r_2+p, \ldots, r_m+p$ and columns $c_1, c_2, \ldots, c_n$.

3. *Find $U \in \mathbb{R}^{r\times m}$ and $V \in \mathbb{R}^{n\times r}$ such that $UMV = I_r$.*

4. *Now determine A, C and G as follows:*

$$\begin{aligned} A &= U\sigma MV, \\ C &= RV, \\ G &= UK. \end{aligned}$$

5. *Find a matrix $P = P^\top \succeq 0$ such that*

$$\left[\begin{array}{c|c} P - APA^\top & G - APC^\top \\ \hline G^\top - CPA^\top & \Lambda(0) - CPC^\top \end{array}\right] \succeq 0, \tag{4.17}$$

6. *Now a linear stochastic realization of the given autocovariances is given by (A, C, P, Q, R, S) where*

$$\begin{aligned} Q &= P - APA^\top, \\ R &= \Lambda(0) - CPC^\top, \\ S &= G - APC^\top. \end{aligned}$$

Notice that Algorithm 4.3 is the linear stochastic equivalent of Algorithm 4.1. It can be proven that there always exists a matrix P that fullfills condition (4.17), hence Algorithm 4.3 always yields a solution to the linear stochastic realization problem.

Proposition 4.13. *Algorithm 4.3 yields a solution to the minimal linear stochastic realization problem of autocovariances Λ.*

As already explained in Section 3.5, for a fixed choice of basis for A, C and G, all solutions to the linear stochastic realization problem are obtained by computing all matrices P that fullfill condition (4.17).

We conclude that the realization problem for linear stochastic models is completely analogous to the realization problem for hidden Markov models.

4.6 Conclusions

In this chapter we considered different versions of the quasi realization problem for hidden Markov models: first the exact quasi realization problem, second the partial quasi realization problem and finally the approximate partial realization problem.

The generalized Hankel matrix of string probabilities is introduced. Some interesting rank properties of the Hankel matrix are proven. We review the realizability question for quasi HMMs. Next, an algorithm is provided to solve the exact quasi realization problem. Several interesting properties of the obtained quasi realization are proven.

We introduce the partial pseudo realization problem. We prove that, if a certain rank condition holds, the minimal pseudo realization algorithm can be solved using the same algorithm as is used for the complete quasi realization problem. In addition under the same rank condition, we prove that a solution to the partial pseudo realization problem is unique up to a similarity transformation. We also give some hints for the solution to the partial pseudo realization problem in case the rank condition does not hold.

The approximate partial pseudo realization problem for hidden Markov models is considered. We propose four different methods to solve this problem. The first three methods aim at obtaining a low rank approximation of the Hankel matrix while the fourth method first builds a full-order balanced realization and subsequently reduces this realization to find an approximate pseudo realization of the string probabilities. We succesfully apply the approximate realization algorithms to the problem of modeling DNA sequences.

Chapter 5

Positive realization of hidden Markov models

The positive realization problem for hidden Markov models was first stated in [20, 52]. The positive realization problem consists in finding a hidden Markov model corresponding to given string probabilities. Analogously to the quasi realization problem, the positive realization problem consists of three subproblems: the first is the realizability problem, the second is the realization problem itself and the third is the equivalence problem. The first two problems are discussed in this chapter and the third problem was considered in Chapter 3.

The positive realization problem is nice from a theoretical point of view. However, in practical applications it is not directly useful because in practise only a finite amount of string probabilities are given and in addition these string probabilities may be estimated instead of exact. The partial realization problem aims at modeling exact string probabilities of strings up to length t. The approximate partial realization problem models approximate string probabilities of strings up to length t.

The positive realization problem for hidden Markov models is closely related to the positive realization problem for linear time-invariant deterministic systems with positive impulse response [5, 29, 99]. The minimal positive realization problem for linear systems has no solution yet. See [12] for a current state of the art of the positive realization problem.

List of own contributions

We here describe our contributions to the positive realization problem for hidden Markov models.

- In Section 5.3.1 we show that the approximate partial Moore realization problem for string probabilities of strings up to length two, can be solved using the structured nonnegative matrix factorization introduced in Section 2.3.

- In Section 5.3.2 the approximate partial Mealy realization problem for string probabilities of strings up to length t is solved by generalizing the Moore realization approach for strings up to length two. Simulation examples show that the methods perform well (Section 5.4).

- In Section 5.5 we apply the approximate realization algorithm to the modeling of the outcome of a coin flipping experiment. Therefore, first strings probabilities are estimated from the sequence and subsequently the approximate realization algorithm is applied to find a model of the sequence.

- In Section 5.6 we apply the approximate realization algorithm to the modeling of DNA sequences. Again, strings probabilities are estimated from the sequences and subsequently the approximate realization algorithm is applied to find a model of the sequences.

Section-by-section overview

In Section 5.1 we review the exact positive realization problem and more precisely the realizability problem and the realization problem itself. In Section 5.2 we review the partial realization problem. In Section 5.3 we consider the approximate partial realization problem. We make a distinction between the Moore and Mealy realization problem. In Section 5.4 we consider two simulation examples showing the effectiveness of the proposed methods. In Section 5.5 we apply the approximate realization algorithm to the problem of modeling the outcome of a coin flipping experiment and in Section 5.6 we apply the approximate realization algorithm to the problem of modeling DNA sequences.

5.1 Exact realization

In this section we give the current status of the exact positive realization problem for Mealy hidden Markov models. First the realizability problem is considered (Section 5.1.1) and subsequently, the realization problem itself (Section 5.1.2) is considered. The results of this section are based on [4, 112, 113].

5.1.1 Realizability problem

The realizability problem is the following: given string probabilities $\mathcal{P}$ for all strings of finite length, derive conditions for the string probabilities to be representable by a positive hidden Markov model. The following theorem [20, 24, 52] contains a necessary condition for string probabilities to be representable by a Mealy hidden Markov model.

Theorem 5.1. *If string probabilities $\mathcal{P}$ are realizable by a Mealy HMM $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ with $|\mathbb{X}| < \infty$,* then *the rank of the generalized Hankel matrix $\mathfrak{H}$ of string probabilities $\mathcal{P}$ is finite.*

It was conjectured in [52] that the condition of Theorem 5.1 is also a sufficient condition for string probabilities to be representable by a Mealy hidden Markov model. However, this conjecture was disproven by showing that there exist examples where the condition holds but where the non-existence of a Mealy hidden Markov realization can be proven [39, 50].

So, what in addition to the condition of Theorem 5.1 is needed to ensure realizability? Several attempts were made to obtain necessary and sufficient conditions for string probabilities to be representable by a Mealy hidden Markov model [34–38]. The most noticable result is the definition of a "cone condition". It is proven that if the process satisfies the cone condition and the associated Hankel matrix has finite rank, then there exists a Mealy hidden Markov realization of the process. However, the cone condition is in essence merely a restatement of the realizability question rather than a solution to it.

In more recent work, Anderson [4] starts with the assumption that the string probabilities have a hidden Markov realization. Hence no conditions for realizability are given in terms of properties of the output process. In [112, 113] sufficient conditions for the realizability of string probabilities are proven. Before being able to recall this sufficient condition we need some definitions from [112, 113].

Definition 5.1. *Consider stationary string probabilities $\mathcal{P}$ of which the associated Hankel matrix $\mathfrak{H}$ has finite rank. Define k as the smallest integer for which*

$$\operatorname{rank} \mathfrak{H}_{(k+1,k+1)} = \operatorname{rank} \mathfrak{H}.$$

Then the string probabilities are called ultra-mixing *if there exists a sequence $d(l)$ that goes to 0 as l goes to infinity such that*

$$\left| \frac{\mathcal{P}(\mathbf{uv})}{\mathcal{P}(\mathbf{v})} - \frac{\mathcal{P}(\mathbf{uvw})}{\mathcal{P}(\mathbf{vw})} \right| \leq d(l), \quad \mathbf{u} \in \mathbb{Y}^k, \mathbf{v} \in \mathbb{Y}^l, \mathbf{w} \in \mathbb{Y}^*.$$

Now, it is proven in [112, 113], that string probabilities $\mathcal{P}$ are realizable by a Mealy HMM $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ with $|\mathbb{X}| < \infty$ *if*

1. The Hankel matrix associated with the string probabilities has finite rank.
2. The string probabilities are alpha-mixing (Definition 4.1).
3. The string probabilities are ultra-mixing (Definition 5.1).
4. A technical condition on the string probabilities is fullfilled (see [112, 113], Theorem 7.2)

On the other hand it is proven in [4] that a Mealy HMM that satisfies another technical condition, generates string probabilities that have finite Hankel rank, are alpha-mixing and are ultra-mixing. Taken together these two results, we have that, modulo two technical conditions, the finite Hankel rank condition, alpha-mixing and ultra-mixing are "almost" necessary and sufficient conditions for string probabilities to be representable by a Mealy hidden Markov model.

5.1.2 Realization problem

The second subproblem of the realization problem for hidden Markov models is the realization problem itself: derive an algorithm that finds a hidden Markov representation of given realizable string probabilities. Up to this moment this problem did not gain much attention in the literature. One of the reasons is probably because the realizability problem has not been solved completely yet. Anderson [4] starts with the assumption that the given string probabilities have a HMM realization and then provides a procedure for constructing a HMM realization. However, there is no guarantee that the HMM constructed using this procedure has the same number of states as the HMM that generated the string probabilities under study.

5.2 Partial realization

The realization problem for hidden Markov models is interesting from theoretical point of view. It supposes that the exact string probabilities of all string of finite length are given. In practical applications however, only a finite number of string probabilities are given. The partial realization problem aims at building a hidden Markov model that models string probabilities of string up to a certain length t. In contradiction to the complete realization problem, the partial realization problem always has a solution. A well-known solution to the partial realization problem is to model the string probabilities with a $(t-1)$-step Markov process. In [114], it is shown that this solution is the only solution that satisfies the conditions:

- The string probabilities for all strings of $\mathbb{Y}^*$ generated by the obtained HMM are consistent.
- The string probabilities for all strings of $\mathbb{Y}^*$ generated by the obtained HMM are nonnegative.

Note that both conditions are trivially fullfilled for string probabilities of strings up to length t. In other words it is proven in [114] that the $(t-1)$-step Markov process is such that the extension of the string probabilities is nonnegative and such that the total string probabilities are consistent.

5.3 Approximate partial realization

In this section we consider the approximate partial realization problem for hidden Markov models. This problem has been considered already in [48]. In the approximate partial realization problem, approximate string probabilities of strings up to length t are given. As the string probabilities are not exact, the solution obtained by the exact partial realization algorithm will typically be of high order. Therefore, it may be more useful to make a good low order approximate realization of the string probabilities rather than to try to match them exactly.

Suppose there is an ordering $(\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_{|\mathbb{Y}|})$ on the symbols from the output set $\mathbb{Y}$, then the approximate string probabilities $\tilde{\mathcal{P}}^{(t)}$ of string of length t can be stacked in the t-dimensional tensor P, defined as

$$P(n_1, n_2, \ldots, n_t) = \tilde{\mathcal{P}}^{(t)}(\mathrm{y}_{n_1}\mathrm{y}_{n_2}\cdots\mathrm{y}_{n_t}).$$

Notice that the element sum of P is equal to 1. As the string probabilities of strings of length smaller than t can be calculated from the string probabilities of strings of length t, the tensor P contains all data for the approximate partial realization problem. From now, we make a distinction between the approximate Moore (Section 5.3.1) and approximate Mealy realization problem (Section 5.3.2).

5.3.1 Moore realization

The approximate t-partial Moore realization problem consists of finding for a given model order, a Moore HMM that approximately generates the string probabilities $\tilde{\mathcal{P}}^{(t)}$. Mathematically, using the tensor P, the approximate partial Moore realization problem becomes:

Problem 5.1 (approximate partial Moore realization problem)**.** *Given the tensor $P \in \mathbb{R}_+^{|\mathbb{Y}|\times|\mathbb{Y}|\times\ldots\times|\mathbb{Y}|}$ of string probabilities of strings of length t, and given the model order $|\mathbb{X}|$, determine $\Pi_\mathbb{X}$, β and $\pi(1)$ of a Moore HMM $(\mathbb{Y}, \mathbb{X}, \Pi_\mathbb{X}, \beta, \pi(1))$, such that $D_{KL}(P, \tilde{P}(\Pi_\mathbb{X}, \beta, \pi(1)))$ is minimized with respect to $\Pi_\mathbb{X}$, β and $\pi(1)$, where $\tilde{P}(\Pi_\mathbb{X}, \beta, \pi(1))$ is defined as*

$$\tilde{P}(\Pi_\mathbb{X}, \beta, \pi(1))(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(t)}) = \pi(1)\,\mathrm{diag}(\beta(\boldsymbol{y}^{(1)}))\Pi_\mathbb{X}\ldots\mathrm{diag}(\beta(\boldsymbol{y}^{(t)}))\Pi_\mathbb{X}e.$$

In case $t = 2$, Problem 5.1 allows a gentle solution using the structured nonnegative matrix factorization introduced in Section 2.3. The Moore realization problem for general t can be solved by extending the techniques of the $t = 2$ case.

In case $t = 2$, the tensor P becomes a matrix of size $|\mathbb{Y}| \times |\mathbb{Y}|$ given by

$$P = \begin{bmatrix} \tilde{\mathcal{P}}^{(2)}(\mathrm{y}_1\mathrm{y}_1) & \tilde{\mathcal{P}}^{(2)}(\mathrm{y}_1\mathrm{y}_2) & \cdots & \tilde{\mathcal{P}}^{(2)}(\mathrm{y}_1\mathrm{y}_{|\mathbb{Y}|}) \\ \tilde{\mathcal{P}}^{(2)}(\mathrm{y}_2\mathrm{y}_1) & \tilde{\mathcal{P}}^{(2)}(\mathrm{y}_2\mathrm{y}_2) & \cdots & \tilde{\mathcal{P}}^{(2)}(\mathrm{y}_2\mathrm{y}_{|\mathbb{Y}|}) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathcal{P}}^{(2)}(\mathrm{y}_{|\mathbb{Y}|}\mathrm{y}_1) & \tilde{\mathcal{P}}^{(2)}(\mathrm{y}_{|\mathbb{Y}|}\mathrm{y}_2) & \cdots & \tilde{\mathcal{P}}^{(2)}(\mathrm{y}_{|\mathbb{Y}|}\mathrm{y}_{|\mathbb{Y}|}) \end{bmatrix}.$$

On the other hand, for a Moore HMM $(\mathbb{Y}, \mathbb{X}, \Pi_\mathbb{X}, \beta, \pi(1))$, it follows from

$$P(y(1) = \mathrm{y}_k, y(2) = \mathrm{y}_l) =$$

$$\sum_{ij} P(y(1) = \mathtt{y}_k, y(2) = \mathtt{y}_l | x(1) = i, x(2) = j) P(x(1) = i, x(2) = j) =$$
$$\sum_{ij} P(y(1) = \mathtt{y}_k | x(1) = i) P(y(2) = \mathtt{y}_l | x(1) = i) P(x(1) = i, x(2) = j) =$$
$$(B^\top \operatorname{diag}(\pi(1)) \Pi_{\mathbb{X}} B)_{kl},$$

that

$$\tilde{P}(\Pi_{\mathbb{X}}, \beta, \pi(1)) = B^\top \operatorname{diag}(\pi(1)) \Pi_{\mathbb{X}} B,$$

where $B = \left[\begin{array}{ccc} \beta(\mathtt{y}_1) & \beta(\mathtt{y}_2) & \ldots \beta(\mathtt{y}_{|\mathbb{Y}|}) \end{array}\right]$.

The problem of finding a Moore HMM of a given order $|\mathbb{X}|$ that approximately realizes string probabilities of strings of length two, is hence equivalent to the problem of finding, for a given $P \in \mathbb{R}_+^{|\mathbb{Y}| \times |\mathbb{Y}|}$ with $e^\top P e = 1$, matrices $B \in \mathbb{R}_+^{|\mathbb{X}| \times |\mathbb{Y}|}$, with $Be = e$, and $\Pi_{\mathbb{X}} \in \mathbb{R}_+^{|\mathbb{X}| \times |\mathbb{X}|}$, with $\Pi_{\mathbb{X}} e = e$, and a vector $\pi(1) \in \mathbb{R}_+^{|\mathbb{X}|}$ with $\pi(1)e = 1$, such that $B^\top \operatorname{diag}(\pi(1)) \Pi_{\mathbb{X}} B$ approximates P optimally with respect to a certain criterion.

In Section 2.3 we developed a method to (approximately) decompose a matrix P into a product $VAV^\top$, with V and A positive, V column stochastic and with the element sum of A equal to the element sum of P. It is easy to see that this method allows us to solve the approximate partial Moore realization problem for strings of length 2. Indeed, from an approximate decomposition $P \simeq VAV^\top$, one can find $\Pi_{\mathbb{X}}$, B and $\pi(1)$ as follows

$$\begin{aligned} B &= V^\top, \\ \Pi_{\mathbb{X}} &= (\operatorname{diag}(Ae))^{-1} A, \\ \pi(1) &= (Ae)^\top. \end{aligned}$$

This defines a Moore HMM that approximately models string probabilities of strings up to length two.

If $y = (y(1), y(2))$ is stationary, i.e. $Pe = P^\top e$, and the decomposition is exact, i.e. $P = B^\top AB$, and B has full row rank, then $x = (x(1), x(2))$ is also stationary, i.e. $Ae = A^\top e$. Indeed, from $Pe = P^\top e$ or $B^\top ABe = B^\top A^\top Be$, we find that $Ae = A^\top e$ if B has full row rank. Note that $\pi(1)$ is a left eigenvector of $\Pi_{\mathbb{X}}$ corresponding to the eigenvalue 1, as is expected for stationary models

$$\begin{aligned} \pi(1)\Pi_{\mathbb{X}} &= (Ae)^\top (\operatorname{diag}(Ae))^{-1} A \\ &= e^\top A = (Ae)^\top \\ &= \pi(1). \end{aligned}$$

So far, we considered the approximate Moore realization problem for string probabilities of strings up to length 2. Two extensions to this problem are possible. The first extension is the Moore realization problem for general string probabilities and the second is the Mealy realization problem for general string probabilities. The approach for both problems is similar. In the next section we concentrate on the most general problem: the approximate Mealy realization problem for general string probabilities.

5.3.2 Mealy realization

In this section we consider the approximate t-partial realization problem for Mealy HMMs. Mathematically, the approximate partial realization problem is defined as floows.

Problem 5.2 (approximate partial realization problem)**.** *Given a tensor* $P \in \mathbb{R}_+^{|\mathbb{Y}|\times|\mathbb{Y}|\times\ldots\times|\mathbb{Y}|}$ *of string probabilities of strings of length* t*, and given the model order* $|\mathbb{X}|$*, determine* Π *and* $\pi(1)$ *of a Mealy HMM* $(\mathbb{Y},\mathbb{X},\Pi,\pi(1))$*, such that* $D_{KL}(P,\tilde{P}(\Pi,\pi(1)))$ *is minimized with respect to* Π *and* $\pi(1)$*, where* $\tilde{P}(\Pi,\pi(1))$ *is defined as*

$$\tilde{P}(\Pi,\pi(1))(\boldsymbol{y}^{(1)},\boldsymbol{y}^{(2)},\ldots,\boldsymbol{y}^{(t)}) = \pi(1)\Pi(\boldsymbol{y}^{(1)})\Pi(\boldsymbol{y}^{(2)})\ldots\Pi(\boldsymbol{y}^{(t)})e.$$

We solve the Mealy realization problem by generalizing the solution to the Moore realization problem for $t = 2$. It is therefore of no coincidence that the theorems of this section are analogous to theorems of Section 2.3 where the structured nonnegative matrix factorization is solved, which is the basis for the Moore realization problem for $t = 2$.

We start with the analogue of Theorem 2.7.

Theorem 5.2. *Let* $P \in \mathbb{R}_+^{|\mathbb{Y}|\times|\mathbb{Y}|\times\ldots\times|\mathbb{Y}|}$*. Then every stationary point* $(\Pi,\pi(1))$ *of the cost function* $D_{\mathrm{KL}}(P,\tilde{P}(\Pi,\pi(1)))$ *satisfies*

$$\begin{aligned}
&\sum_{y_2,y_3,\cdots y_t} P(\boldsymbol{y},y_2,y_3,\ldots,y_t)+\\
&\qquad\sum_{y_1,y_3,y_4\cdots y_t} P(y_1,\boldsymbol{y},y_3,y_4,\ldots,y_t)+\ldots+\\
&\qquad\qquad\sum_{y_1,y_2,\cdots y_{t-1}} P(y_1,y_2,\ldots,y_{t-1},\boldsymbol{y}) =\\
&\sum_{y_2,y_3,\cdots y_t} \tilde{P}(\Pi,\pi(1))(\boldsymbol{y},y_2,y_3,\ldots,y_t)+\\
&\qquad\sum_{y_1,y_3,y_4\cdots y_t} \tilde{P}(\Pi,\pi(1))(y_1,\boldsymbol{y},y_3,y_4,\ldots,y_t)+\ldots+\\
&\qquad\qquad\sum_{y_1,y_2,\cdots y_{t-1}} \tilde{P}(\Pi,\pi(1))(y_1,y_2,\ldots,y_{t-1},\boldsymbol{y}), \quad \boldsymbol{y}\in\mathbb{Y}.
\end{aligned}$$

As a consequence the total sum of P *and* $\tilde{P}$ *are equal*

$$\sum_{y_1,y_2,\cdots y_t} P(y_1,y_2,\ldots,y_t) = \sum_{y_1,y_2,\cdots y_t} \tilde{P}(\Pi,\pi(1))(y_1,y_2,\ldots,y_t).$$

Proof: The proof is analogous to the proof of Theorem 2.7. ■

We call a stationary point of the divergence $D(P,\tilde{P}(\Pi,\pi(1)))$ *normalized*, if it holds that $\pi(1)e = 1$ and $\sum_{\mathrm{y}}\Pi(\mathrm{y})e = e$. Only normalized points give rise to consistent hidden Markov models, so we restrict the search for an optimal solution to Problem 5.2 to the space of normalized points. Restricting the search to normalized points can be done by chosing normalized initial values and by making sure that the update formulas retain the normalization. Chosing normalized initial values is no problem. The point is to find update formulas that retain the normalization. The following theorem is the analogue of Theorem 2.8.

Theorem 5.3. *Assume that the starting values* $\Pi^{(0)}$ *and* $\pi^{(0)}(1)$ *are normalized, i.e.* $\pi^{(0)}(1)e = 1$ *and* $\sum_{\boldsymbol{y}} \Pi^{(0)}(\boldsymbol{y})e = e$. *Then the divergence* $D_{KL}(P||\tilde{P}(\Pi, \pi(1)))$ *is nonincreasing under the update rules*

$$\pi_i^{(k+1)}(1) = \pi_i^{(k)}(1) \sum_{y_1, y_2, \ldots, y_t} \frac{P(y_1, \ldots, y_t)}{\tilde{P}(\Pi, \pi^{(k)}(1))(y_1, \ldots, y_t)} \Pi_{i,:}(y_1)\Pi(y_2)\ldots\Pi(y_t)e \tag{5.1}$$

$$\Pi_{ij}^{(k+1)}(\boldsymbol{y}) = \Pi_{ij}^{(k)}(\boldsymbol{y}) \frac{A + B + \ldots + C}{D + E + \ldots + F}, \tag{5.2}$$

where

$$\begin{aligned}
A &= \sum_{y_2 y_3 \cdots y_t} \frac{P(\boldsymbol{y}, y_2, \ldots, y_t)}{\tilde{P}(\Pi^{(k)}, \pi(1))(\boldsymbol{y}, y_2, \ldots, y_t)} \cdot \\
&\qquad \pi_i(1)\Pi_{j,:}^{(k)}(y_2)\Pi^{(k)}(y_3)\ldots\Pi^{(k)}(y_t)e, \\
B &= \sum_{y_1 y_3 y_4 \cdots y_t} \frac{P(y_1, \boldsymbol{y}, y_3, \ldots, y_t)}{\tilde{P}(\Pi^{(k)}, \pi(1))(y_1, \boldsymbol{y}, y_3, \ldots, y_t)} \cdot \\
&\qquad \pi(1)\Pi_{:,i}^{(k)}(y_1)\Pi_{j,:}^{(k)}(y_3)\Pi^{(k)}(y_4)\ldots\Pi^{(k)}(y_t)e, \\
C &= \sum_{y_1 y_2 \cdots y_{t-1}} \frac{P(y_1, \ldots, y_{t-1}, \boldsymbol{y})}{\tilde{P}(\Pi^{(k)}, \pi(1))(y_1, \ldots, y_{t-1}, \boldsymbol{y})} \cdot \\
&\qquad \pi(1)\Pi^{(k)}(y_1)\ldots\Pi^{(k)}(y_{t-2})\Pi_{:,i}^{(k)}(y_{t-1}), \\
D &= \sum_{y_1 y_2 \cdots y_t} \frac{P(y_1, y_2, \ldots, y_t)}{\tilde{P}(\Pi^{(k)}, \pi(1))(y_1, y_2, \ldots, y_t)} \cdot \\
&\qquad \pi_i(1)\Pi_{i,:}^{(k)}(y_1)\Pi^{(k)}(y_2)\ldots\Pi^{(k)}(y_t)e, \\
E &= \sum_{y_1 y_2 \cdots y_t} \frac{P(y_1, y_2, \ldots, y_t)}{\tilde{P}(\Pi^{(k)}, \pi(1))(y_1, y_2, \ldots, y_t)} \cdot \\
&\qquad \pi(1)\Pi_{:,i}^{(k)}(y_1)\Pi_{i,:}^{(k)}(y_2)\Pi^{(k)}(y_3)\ldots\Pi^{(k)}(y_t)e, \\
F &= \sum_{y_1 y_2 \cdots y_t} \frac{P(y_1, y_2, \ldots, y_t)}{\tilde{P}(\Pi^{(k)}, \pi(1))(y_1, y_2, \ldots, y_t)} \cdot \\
&\qquad \pi(1)\Pi^{(k)}(y_1)\ldots\Pi_{:,i}^{(k)}(y_{t-1})\Pi_{i,:}^{(k)}(y_t)e.
\end{aligned} \tag{5.3}$$

In addition, the updated values of Π *and* $\pi(1)$ *are also normalized.*

Proof: The proof is analogous to the proof of Theorem 2.8. ■

For fixed $\Pi^{(k)} = \Pi^{(0)}$, the divergence $F(\Pi, \pi(1)) = D_{KL}(P||\tilde{P}(\Pi, \pi(1))$ is invariant under an update of $\pi(1)$, i.e. $\pi^{(k+1)}(1) = \pi^{(k)}(1)$, if and only if $\pi^{(k)}(1)$ is a stationary point of the divergence with fixed $\Pi = \Pi^{(0)}$, i.e. $\pi_i^{(k)}(1)\frac{\partial F}{\partial \pi_i(1)}(\pi^{(k)}(1), \Pi^{(0)}) = 0$. For fixed $\pi^{(k)}(1) = \pi^{(0)}(1)$ on the other hand, the divergence is invariant under an update for Π, i.e. $\Pi^{(k+1)} = \Pi^{(k)}$, if and only if

$$\Pi(\mathrm{y})_{ij}^{(k)} (D + E + F - A - B - C) \quad = \quad 0,$$

where A, B, C, D, E, F are defined in (5.3). Notice that this last condition is in general not equivalent to the condition that $\Pi^{(k)}$ is a stationary point of the divergence with fixed $\pi(1) = \pi^{(0)}(1)$. So for the case where we take $\pi(1)$ fixed and update only Π, it is possible that the formulas, if they converge, converge to a point that is not a stationary point of the divergence with fixed $\pi(1)$.

However, if we use the update formulas for $\pi(1)$ and Π alternatingly, i.e.

$$(\pi^{(0)}(1), \Pi^{(0)}) \mapsto (\pi^{(1)}(1), \Pi^{(0)}) \mapsto (\pi^{(1)}(1), \Pi^{(1)}) \mapsto (\pi^{(2)}(1), \Pi^{(1)}) \mapsto \ldots,$$

we have the following result analogous to Theorem 2.9.

Theorem 5.4. *The divergence is invariant under updates (5.1) and (5.2) if and only if* $(\Pi, \pi(1))$ *is a stationary point of the divergence, i.e.*

$$\begin{cases} \pi^{(k+1)}(1) = \pi^{(k)}(1), \\ \Pi^{(k+1)} = \Pi^{(k)}, \end{cases}$$

$$\Leftrightarrow \quad \begin{cases} \pi_i^{(k)}(1)\frac{\partial F}{\partial \pi_i}(\Pi^{(k)}, \pi^{(k)}(1)) = 0, & i = 1, 2, \ldots |\mathbb{X}|, \\ \Pi_{ij}^{(k)}(\boldsymbol{y})\frac{\partial F}{\partial \Pi_{ij}(\boldsymbol{y})}(\Pi^{(k)}, \pi^{(k)}(1)) = 0, & i, j = 1, 2, \ldots |\mathbb{X}|, \boldsymbol{y} \in \mathbb{Y}. \end{cases}$$

Proof: The proof is analogous to the proof of Theorem 2.9. ■

It follows that if the update formulas converge, that they converge to a stationary point of the cost function in case $\pi(1)$ and Π are updated alternatingly or in case Π is fixed and $\pi(1)$ is updated. However, when $\pi(1)$ is fixed and only Π is updated, it is only guarantueed that the divergence is nonincreasing. It is possible that the formulas converge to a point that is not a stationary point of the divergence with fixed $\pi(1)$.

The algorithm below implements the same update formulas as in Theorem 5.3, but is better from computational point of view as the denominator of Equation (5.2) does not have to be computed

Algorithm 5.1. *Start with random nonnegative* $\pi^{(0)}(1)$ *and* $\Pi^{(0)}$ *with* $\pi^0(1)e = 1$ *and* $\sum_y \Pi^{(0)}(y)e = e$*, iterate the following steps for* $k = 0, 1, ...$ *until convergence:*

1. $\pi_i^{(k+1)}(1) = \pi_i^{(k)}(1)\sum_{y_1, y_2, \ldots, y_t} \frac{P(y_1, \ldots, y_t)}{\tilde{P}(\Pi, \pi^{(k)}(1))(y_1, \ldots, y_t)}\Pi_{i,:}(y_1)\Pi(y_2)\ldots\Pi(y_t)e$

2. $\Pi_{ij}^{(k+1)}(\boldsymbol{y}) = \Pi_{ij}^{(k)}(\boldsymbol{y})(A + B + \ldots + C)$ *with*

$$\begin{aligned}
A &= \sum_{y_2 y_3 \cdots y_t} \frac{P(\boldsymbol{y}, y_2, \ldots, y_t)}{\tilde{P}(\Pi^{(k)}, \pi(1))(\boldsymbol{y}, y_2, \ldots, y_t)} \cdot \\
&\qquad \pi_i(1)\Pi_{j,:}^{(k)}(y_2)\Pi^{(k)}(y_3)\ldots\Pi^{(k)}(y_t)e, \\
B &= \sum_{y_1 y_3 y_4 \cdots y_t} \frac{P(y_1, \boldsymbol{y}, y_3, \ldots, y_t)}{\tilde{P}(\Pi^{(k)}, \pi(1))(y_1, \boldsymbol{y}, y_3, \ldots, y_t)} \cdot \\
&\qquad \pi(1)\Pi_{:,i}^{(k)}(y_1)\Pi_{j,:}^{(k)}(y_3)\Pi^{(k)}(y_4)\ldots\Pi^{(k)}(y_t)e, \\
C &= \sum_{y_1 y_2 \cdots y_{t-1}} \frac{P(y_1, \ldots, y_{t-1}, \boldsymbol{y})}{\tilde{P}(\Pi^{(k)}, \pi(1))(y_1, \ldots, y_{t-1}, \boldsymbol{y})} \cdot \\
&\qquad \pi(1)\Pi^{(k)}(y_1)\ldots\Pi^{(k)}(y_{t-2})\Pi_{:,i}^{(k)}(y_{t-1}).
\end{aligned}$$

3. Normalize $\Pi^{(k+1)}$ *such that* $\sum_{\boldsymbol{y}} \Pi^{(k+1)}(\boldsymbol{y})e = e$.

5.4 Simulation example

In this section we describe two simulation examples to show the effectiveness of the proposed methods. The first simulation example builds a Moore model for string probabilities of strings up to length 2 (Section 5.4.1) and the second simulation example builds a Mealy model for string probabilities of strings up to length 3 (Section 5.4.2).

5.4.1 Moore realization, $t = 2$

Suppose we are given the probabilities of strings of length two of a Moore HMM with $\mathbb{Y} = \{a, b, \ldots, j\}$. The objective is to find the system matrices of the underlying Moore HMM. Suppose there is an ordering ($\mathrm{y}_1 = a, \mathrm{y}_2 = b, \ldots, \mathrm{y}_{10} = j$) on the output set $\mathbb{Y}$. Now the string probabilities can be stacked in the matrix P as described before, i.e. $P_{kl} = \mathcal{P}(\mathrm{y}_k \mathrm{y}_l)$.

In our simulation example P is given by

$$P = \begin{bmatrix}
396 & 193 & 149 & 116 & 113 & 94 & 98 & 161 & 128 & 454 \\
182 & 128 & 87 & 85 & 77 & 67 & 70 & 120 & 84 & 191 \\
150 & 87 & 69 & 60 & 58 & 52 & 53 & 77 & 63 & 150 \\
111 & 84 & 60 & 61 & 55 & 51 & 52 & 80 & 57 & 112 \\
112 & 75 & 58 & 55 & 51 & 47 & 48 & 70 & 54 & 105 \\
92 & 67 & 50 & 51 & 46 & 45 & 45 & 63 & 47 & 93 \\
97 & 69 & 52 & 52 & 47 & 46 & 46 & 65 & 49 & 96 \\
149 & 118 & 78 & 80 & 72 & 63 & 65 & 114 & 78 & 148 \\
126 & 81 & 64 & 58 & 55 & 49 & 51 & 75 & 60 & 113 \\
488 & 189 & 152 & 105 & 100 & 86 & 90 & 141 & 111 & 415
\end{bmatrix} 10^{-4}.$$

This matrix P was generated as $P = B^\top \operatorname{diag}(\pi(1))\Pi_{\mathbb{X}}B$ where B, $\Pi_{\mathbb{X}}$ and $\pi(1)$ are the system matrices of a stationary Moore HMM $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, B, \pi(1))$ with $\mathbb{X} = \{1, 2, \ldots, 5\}$ and

$$\begin{aligned}
\Pi_{\mathbb{X}} &= \begin{bmatrix} 0.80 & 0.00 & 0.10 & 0.10 & 0.00 \\ 0.20 & 0.20 & 0.20 & 0.20 & 0.20 \\ 0.40 & 0.10 & 0.30 & 0.20 & 0.00 \\ 0.15 & 0.05 & 0.10 & 0.35 & 0.35 \\ 0.05 & 0.05 & 0.05 & 0.55 & 0.30 \end{bmatrix}, \\
\pi(1) &= \begin{bmatrix} 0.4850 & 0.0375 & 0.1218 & 0.2300 & 0.1257 \end{bmatrix}, \\
B &= \begin{bmatrix} 0.10 & 0.10 & 0.10 & 0.10 & 0.10 & 0.10 & 0.10 & 0.10 & 0.10 & 0.10 \\ 0.15 & 0.00 & 0.25 & 0.00 & 0.20 & 0.00 & 0.05 & 0.00 & 0.35 & 0.00 \\ 0.30 & 0.30 & 0.00 & 0.10 & 0.00 & 0.00 & 0.00 & 0.30 & 0.00 & 0.00 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.55 \\ 0.70 & 0.10 & 0.10 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.10 \end{bmatrix}.
\end{aligned}$$

In the simulation example, this model is unknown, but we give it here to check the performance of the algorithms.

We use the iterative update algorithm of Theorem 2.8 to compute optimal approximations with respect to the Kullback-Leibler divergence with system order equal to $1, 2, \ldots, 10$. As initial values for the iterative algorithm we use randomly chosen nonnegative matrices. As stopping rule, we use the Kullback-Leibler divergence between the approximation at iteration step t and the approximation at step $t+1$. The algorithm stops if this divergence is smaller than 10^{-8}. In Table 5.1 we show the number of steps until convergence for the different system orders.

On Figure 5.1, we plot the Kullback-Leibler divergence between the original matrix P and its optimal approximation with respect to the Kullback-Leibler divergence as a function of the system order.

Table 5.1: *Number of iterations for the multiplicative update method minimizing the Kullback-Leibler divergence.*

sp-rank	1	2	3	4	5	6	7	8	9	10
number of iterations	1	272	1439	1431	2137	3656	2157	2320	1786	1806

Notice that the divergence is almost equal to 0 for system order 5 to 10. This makes sense as the matrix P was generated using an underlying hidden Markov model of order 5. To show further the quality of the approximations, we give in Table 5.2 the true output probabilities of a selection of strings of length 2 and compare them with the probabilities found with the Kullback-Leibler minimalisation method of order $5, 4, \ldots 1$. We conclude that the modeling of string probabilities of strings of length 2 with a hidden Markov model works well.

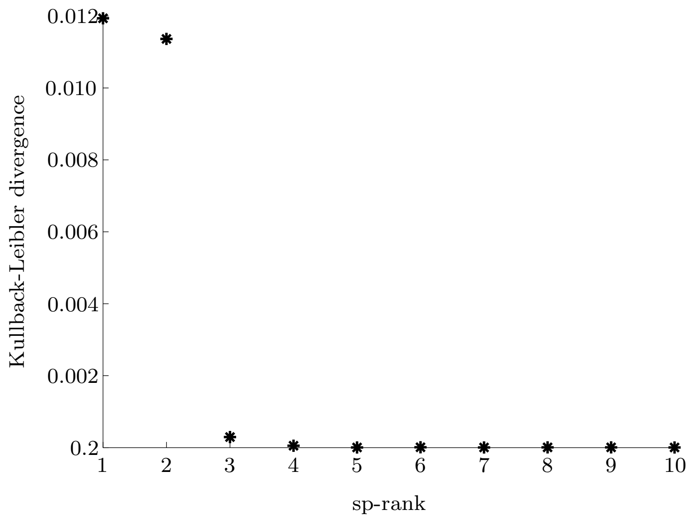


Figure 5.1: *Kullback-Leibler divergence between the true matrix P and its optimal (w.r.t. the Kullback-Leibler divergence) approximation for system order $1, 2, \ldots, 10$ computed with the iterative algorithm of Theorem 2.8.*

Table 5.2: *String probabilities for strings of length 2.*

Sequence	Exact	Order 5	Order 4	Order 3	Order 2	Order 1
aa	0.0396	0.0397	0.0396	0.0397	0.0333	0.0362
ab	0.0193	0.0192	0.0192	0.0190	0.0204	0.0207
ac	0.0149	0.0149	0.0149	0.0150	0.0153	0.0156
ad	0.0116	0.0116	0.0116	0.0116	0.0137	0.0137
ae	0.0113	0.0113	0.0113	0.0114	0.0131	0.0128
af	0.0094	0.0094	0.0094	0.0095	0.0113	0.0114
ag	0.0098	0.0098	0.0099	0.0100	0.0118	0.0118
ah	0.0161	0.0161	0.0161	0.0158	0.0185	0.0184
ai	0.0128	0.0128	0.0127	0.0127	0.0144	0.0139
aj	0.0454	0.0454	0.0454	0.0454	0.0384	0.0357

5.4.2 Mealy realization, $t = 3$

Suppose we are given the probabilities of strings of length three of a process with $\mathbb{Y} = \{a, b, c, d\}$ generated by an unknown Mealy hidden Markov model with $|\mathbb{X}| = 3$. The objective is to find an approximate Mealy HMM corresponding to the string probabilities. Suppose there is an ordering $(\mathrm{y}_1 = a, \mathrm{y}_2 = b, \mathrm{y}_2 = c, \mathrm{y}_2 = d)$ on the output set $\mathbb{Y}$. Now the string probabilities can be stacked in the tensor P as described before, i.e. $P(k, l, m) = \mathcal{P}(\mathrm{y}_k \mathrm{y}_l \mathrm{y}_m)$. In this simulation example, the tensor P is given by

$$
\begin{aligned}
P(:,:,1) &= \begin{bmatrix} 0.0372 & 0.0093 & 0.0118 & 0.0015 \\ 0.0083 & 0.0020 & 0.0070 & 0.0023 \\ 0.0188 & 0.0047 & 0.0325 & 0.0015 \\ 0.0072 & 0.0013 & 0.0158 & 0.0096 \end{bmatrix}, \\
P(:,:,2) &= \begin{bmatrix} 0.0170 & 0.0045 & 0.0056 & 0.0009 \\ 0.0038 & 0.0015 & 0.0033 & 0.0043 \\ 0.0087 & 0.0036 & 0.0161 & 0.0021 \\ 0.0033 & 0.0027 & 0.0073 & 0.0192 \end{bmatrix}, \\
P(:,:,3) &= \begin{bmatrix} 0.0164 & 0.0173 & 0.0382 & 0.0108 \\ 0.0051 & 0.0037 & 0.0229 & 0.0061 \\ 0.0344 & 0.0088 & 0.2368 & 0.0062 \\ 0.0031 & 0.0027 & 0.0173 & 0.0192 \end{bmatrix}, \\
P(:,:,4) &= \begin{bmatrix} 0.0072 & 0.0045 & 0.0036 & 0.0028 \\ 0.0017 & 0.0057 & 0.0022 & 0.0317 \\ 0.0047 & 0.0148 & 0.0153 & 0.0141 \\ 0.0014 & 0.0199 & 0.0035 & 0.1437 \end{bmatrix}.
\end{aligned}
$$

Using Algorithm 5.1, we build a Mealy model of order 1,2,3,4 and 5. The Kullback-Leibler divergence between P and $\tilde{P}(\Pi, \pi(1))$ for the different models is given in Table 5.4.

Table 5.3: *Kullback-Leibler divergence between the tensors P and $\tilde{P}(\Pi, \pi(1))$ for a Mealy hidden Markov model of order* 1,2,3,4 *and* 5.

order	1	2	3	4	5
KL-divergence	0.5390	0.1216	0.0038	0.0036	0.0035

It is clear that order 3 gives the best trade-off between accuracy and complexity of the model. The model of order three obtained by to algorithm is given by $(\mathbb{X}, \mathbb{X}, \Pi, \pi(1))$, with $\mathbb{X} = \{1, 2, 3\}$ and

$$
\begin{aligned}
\Pi(a) &= \begin{bmatrix} 0.4801 & 0.0106 & 0.0013 \\ 0.0102 & 0.0699 & 0.0000 \\ 0.0190 & 0.0001 & 0.0002 \end{bmatrix}, \\
\Pi(b) &= \begin{bmatrix} 0.1021 & 0.0933 & 0.0234 \\ 0.0019 & 0.0008 & 0.0382 \\ 0.0011 & 0.0001 & 0.0934 \end{bmatrix}, \\
\Pi(c) &= \begin{bmatrix} 0.0716 & 0.1258 & 0.0056 \\ 0.0746 & 0.7598 & 0.0120 \\ 0.0808 & 0.0001 & 0.0040 \end{bmatrix},
\end{aligned}
$$

$$\begin{aligned}\Pi(d) &= \begin{bmatrix} 0.0134 & 0.0573 & 0.0155 \\ 0.0005 & 0.0034 & 0.0287 \\ 0.0437 & 0.0124 & 0.7451 \end{bmatrix}, \\ \pi(1) &= \begin{bmatrix} 0.3049 & 0.3968 & 0.2983 \end{bmatrix}.\end{aligned}$$

The tensor $\tilde{P}(\Pi, \pi(1))$ is given by

$$\begin{aligned}\tilde{P}(\Pi,\pi(1))(:,:,1) &= \begin{bmatrix} 0.0374 & 0.0091 & 0.0101 & 0.0018 \\ 0.0084 & 0.0020 & 0.0063 & 0.0022 \\ 0.0216 & 0.0050 & 0.0370 & 0.0015 \\ 0.0066 & 0.0016 & 0.0125 & 0.0090 \end{bmatrix}, \\ \tilde{P}(\Pi,\pi(1))(:,:,2) &= \begin{bmatrix} 0.0167 & 0.0046 & 0.0048 & 0.0012 \\ 0.0038 & 0.0015 & 0.0030 & 0.0043 \\ 0.0097 & 0.0036 & 0.0182 & 0.0021 \\ 0.0029 & 0.0028 & 0.0058 & 0.0194 \end{bmatrix}, \\ \tilde{P}(\Pi,\pi(1))(:,:,3) &= \begin{bmatrix} 0.0185 & 0.0160 & 0.0394 & 0.0084 \\ 0.0054 & 0.0038 & 0.0237 & 0.0060 \\ 0.0289 & 0.0092 & 0.2334 & 0.0065 \\ 0.0042 & 0.0038 & 0.0210 & 0.0207 \end{bmatrix}, \\ \tilde{P}(\Pi,\pi(1))(:,:,4) &= \begin{bmatrix} 0.0068 & 0.0058 & 0.0036 & 0.0034 \\ 0.0016 & 0.0056 & 0.0022 & 0.0313 \\ 0.0043 & 0.0133 & 0.0151 & 0.0136 \\ 0.0012 & 0.0191 & 0.0036 & 0.1440 \end{bmatrix}.\end{aligned}$$

The tensor $\tilde{P}(\Pi, \pi(1))$ is not exactly equal to P as the decomposition algorithm converges only to a local optimum of the cost function $D_{KL}(P, \tilde{P}(\Pi, \pi(1)))$. However, $\tilde{P}(\Pi, \pi(1))$ is close to P, from which we conclude that the modeling works well.

5.5 Modeling a coin flipping sequence

In this section we apply the positive realization method to a coin flipping experiment. The setup is the following: Person A and Person B are flipping coins. If the result is a `head`, the score of person A is increased with one. If the result is a `tail`, the score of Person B is increased with one. After a sequence of 100 experiments, the person with the highest score wins the game. Person A does the tossing, however, he does not play the game fairly. He starts tossing with a fair coin with the following probabilities

$$P_{\text{fair}}(\texttt{head}) = \tfrac{1}{2}, \qquad P_{\text{fair}}(\texttt{tail}) = \tfrac{1}{2}.$$

However, when Person B is earning too much, Person A switches to a false coin with probabilities

$$P_{\text{false}}(\texttt{head}) = \tfrac{5}{6}, \qquad P_{\text{false}}(\texttt{tail}) = \tfrac{1}{6}.$$

More precisely, Person A switches to the false coin if the outcome of the two last experiments was `tail`. To make sure that Person B does not become suspicious, Person A switches back to the fair coin after two consecutive `heads`. Person C notices that Person A is not playing fair but does not know the "rule" that Person A uses to switch from the one coin to the other. Person C is able to write down the outcomes of a sequence of 100 flipping experiments as well as the coin (`fair` or `false`) that was used for each of the experiments. For a second sequence of experiments, Person C is only able to see the outcomes of the experiments, but he can not determine which coin was used. He wants to find out at which time instants the fair coin was used and at which time instants the false coin was used. Therefore, he first builds a model of the sequence and subsequently applies estimation techniques to find out which coin was used. In this section we describe how to build the model and in Section 7.5 we explain how to find out which coin was used in the second sequence of experiments.

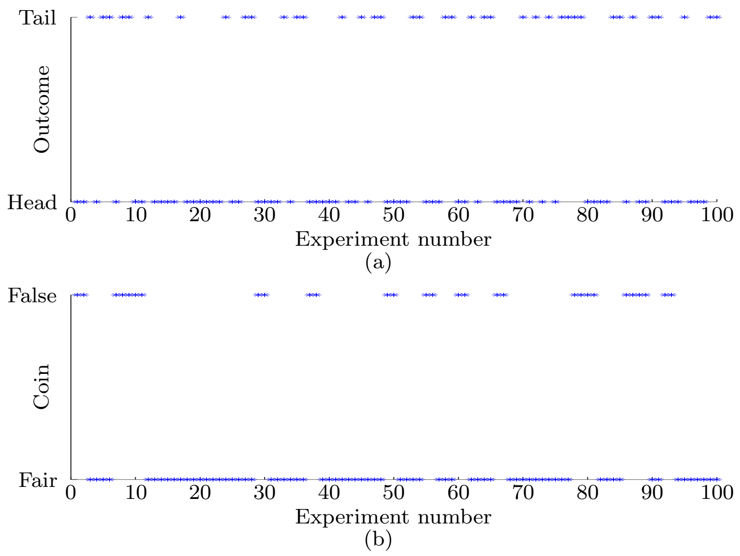


Figure 5.2: *In Subfigure (a), we show the outcomes of the given sequence of coin flipping experiments. In Subfigure (b), we show which coin was used for each of the experiments.*

In Figure 5.2(a), we show the outcomes of the first sequence of experiments and in Figure 5.2(b), we show the coin that was used for each experiment. We want to make a joint model of both sequences, the first taking values in $\mathbb{Y} = \{\texttt{head}, \texttt{tail}\}$ and the second in $\mathbb{Z} = \{\texttt{fair}, \texttt{false}\}$. We first need to define hidden Markov models with two output processes. A Mealy HMM with output processes y and z is defined as $(\mathbb{X}, \mathbb{Y} \times \mathbb{Z}, \Pi, \pi(1))$. Notice that the output alphabet of the process z is denoted by $\mathbb{Z}$ (not to be confused with the set of integers). Notice that Π is hence a mapping of the form $\Pi : \mathbb{Y} \times \mathbb{Z} \mapsto \mathbb{R}_+^{|\mathbb{X}| \times |\mathbb{X}|}$. A hidden Markov model with two output processes can be converted into a hidden Markov model with one output process $(\mathbb{X}, \mathbb{W}, \Pi, \pi(1))$ by taking $\mathbb{W} = \mathbb{Y} \times \mathbb{Z}$. By doing so, all methods developped for HMMs with one output process, can be used for HMMs with two output processes. In this example we use $\mathbb{W} = \{a, b, c, d\}$ with

$$\begin{aligned} a &= \texttt{head},\ \texttt{fair}, \\ b &= \texttt{tail},\ \texttt{fair}, \\ c &= \texttt{head},\ \texttt{false}, \\ d &= \texttt{tail},\ \texttt{false}. \end{aligned}$$

Now the string probabilities of strings of length 3 are estimated using Equation (3.3) and subsequently the positive realization procedure proposed in this chapter is applied to obtain a model. The estimated string probabilities of strings up to length 3, stacked in the tensor P, are given by

$$\begin{aligned}
P(:,:,1) &= \begin{bmatrix} 0.0888 & 0.0867 & 0 & 0 \\ 0.0577 & 0 & 0 & 0 \\ 0.0303 & 0.0302 & 0.0573 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\
P(:,:,2) &= \begin{bmatrix} 0.0880 & 0.0875 & 0 & 0 \\ 0.0591 & 0 & 0 & 0 \\ 0.0270 & 0.0269 & 0.0571 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\
P(:,:,3) &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.0957 & 0.0784 & 0.0155 \\ 0 & 0 & 0 & 0.0196 \\ 0 & 0 & 0.0360 & 0.0073 \end{bmatrix}, \\
P(:,:,4) &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.0187 & 0.0173 & 0.0032 \\ 0 & 0 & 0 & 0.0041 \\ 0 & 0 & 0.0064 & 0.0009 \end{bmatrix}.
\end{aligned}$$

Using Algorithm 5.1, we build a Mealy model of order 1,2,3,4, 5 and 6. The Kullback-Leibler divergence between P and $\tilde{P}$ for the different models is given in Table 5.4.

Table 5.4: *Kullback-Leibler divergence between the tensors P and $\tilde{P}(\Pi, \pi(1))$ for a Mealy hidden Markov model of order* 1,2,3,4, 5 *and* 6.

order	1	2	3	4	5	6
KL-divergence	0.8133	0.3977	0.1760	0.0060	0.0058	0.0058

It is clear that order 4 gives the best trade-off between accuracy and complexity of the model. The model of order 4 obtained by the algorithm is given by $(\mathbb{X}, \mathbb{Y} \times \mathbb{Z}, \Pi, \pi(1))$, with $\mathbb{X} = \{1, 2, 3, 4\}$ and

$$\begin{aligned}
\Pi(\texttt{head}, \texttt{fair}) &= \begin{bmatrix} 0.0001 & 0.4781 & 0.0001 & 0.0000 \\ 0.0001 & 0.4913 & 0.0003 & 0.0002 \\ 0.0000 & 0.0002 & 0.0001 & 0.0001 \\ 0.0000 & 0.0000 & 0.0006 & 0.0007 \end{bmatrix}, \\
\Pi(\texttt{tail}, \texttt{fair}) &= \begin{bmatrix} 0.0018 & 0.0000 & 0.0003 & 0.5168 \\ 0.4934 & 0.0140 & 0.0000 & 0.0000 \\ 0.0005 & 0.0000 & 0.0000 & 0.0000 \\ 0.0007 & 0.0000 & 0.0000 & 0.0001 \end{bmatrix}, \\
\Pi(\texttt{head}, \texttt{false}) &= \begin{bmatrix} 0.0000 & 0.0000 & 0.0021 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0055 & 0.8314 & 0.0006 & 0.0000 \\ 0.0000 & 0.0017 & 0.8232 & 0.0057 \end{bmatrix}, \\
\Pi(\texttt{tail}, \texttt{false}) &= \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0007 \\ 0.0005 & 0.0000 & 0.0000 & 0.0002 \\ 0.0002 & 0.0000 & 0.0003 & 0.1611 \\ 0.0016 & 0.0000 & 0.0008 & 0.1649 \end{bmatrix}, \\
\pi(1) &= \begin{bmatrix} 0.2305 & 0.4558 & 0.1426 & 0.1712 \end{bmatrix}.
\end{aligned}$$

The tensor $\tilde{P}(\Pi, \pi(1))$ corresponding to $(\mathbb{X}, \mathbb{Y} \times \mathbb{Z}, \Pi, \pi(1))$ is given by

$$\begin{aligned}
\tilde{P}(\Pi, \pi(1))(:, :, 1) &= \begin{bmatrix} 0.0808 & 0.0812 & 0.0001 & 0.0001 \\ 0.0546 & 0.0019 & 0.0002 & 0.0001 \\ 0.0289 & 0.0289 & 0.0582 & 0.0001 \\ 0.0001 & 0.0000 & 0.0001 & 0.0001 \end{bmatrix}, \\
\tilde{P}(\Pi, \pi(1))(:, :, 2) &= \begin{bmatrix} 0.0833 & 0.0879 & 0.0001 & 0.0001 \\ 0.0563 & 0.0020 & 0.0002 & 0.0001 \\ 0.0298 & 0.0313 & 0.0601 & 0.0001 \\ 0.0001 & 0.0000 & 0.0001 & 0.0000 \end{bmatrix}, \\
\tilde{P}(\Pi, \pi(1))(:, :, 3) &= \begin{bmatrix} 0.0001 & 0.0004 & 0.0002 & 0.0001 \\ 0.0001 & 0.0969 & 0.0831 & 0.0165 \\ 0.0001 & 0.0005 & 0.0008 & 0.0191 \\ 0.0001 & 0.0002 & 0.0357 & 0.0071 \end{bmatrix}, \\
\tilde{P}(\Pi, \pi(1))(:, :, 4) &= \begin{bmatrix} 0.0001 & 0.0001 & 0.0000 & 0.0000 \\ 0.0001 & 0.0195 & 0.0160 & 0.0033 \\ 0.0001 & 0.0001 & 0.0002 & 0.0039 \\ 0.0000 & 0.0000 & 0.0069 & 0.0014 \end{bmatrix}.
\end{aligned}$$

It is clear that $\tilde{P}(\Pi, \pi(1))$ is a good approximation for P. In Section 7.5, the model obtained in this section is used for finding out which coin was used in a sequence of flipping experiments. The goods results of that section are another proof that the model is of high quality.

5.6 Modeling DNA sequences

In this section we apply the approximate partial realization algorithm to the modeling of DNA sequences. We consider the same problem setting as in Section 4.4, but now we compute a positive hidden Markov model instead of a quasi hidden Markov model of the given output sequences.

In Section 4.4 we were given 40 DNA sequences $\mathbf{u}_1, \ldots, \mathbf{u}_{40}$ of length 200 and the goal was to find a quasi hidden Markov model of the 40 strings. Therefore, we first calculated the string probabilities of strings up to length 4 and subsequently, we applied the approximate partial quasi realization algorithm of Section 4.3.3 to find an approximate quasi hidden Markov model. In this section we make use of the string probabilities of strings of length 4, but now we use Algorithm 5.1 to build a positive instead of a quasi hidden Markov model. Analogous to Section 4.4 we build models of order 1, 2, 3, 4, 5 and 6. The algorithms converged (change in KL-divergence is smaller than 0.00001) after 20 steps approximately. The Kullback-Leibler divergence between the Hankel block $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(3,3)}$ and the corresponding block of the Hankel matrix generated by the obtained model is given in Table 5.5. We also repeat the corresponding distances for quasi models (see Table 5.5).

Table 5.5: *Kullback-Leibler divergence between the Hankel block $(\mathfrak{H}_{\tilde{\mathcal{P}}})_{(3,3)}$ and the corresponding block of the Hankel matrix generated by the obtained Mealy hidden Markov model of order* 1,2,3,4, 5 *and* 6 *obtained using the approximate quasi realization procedure and the approximate positive realization procedure.*

order	1	2	3	4	5	6	7
quasi HMM	0.1109	0.0653	0.0449	0.0263	0.0220	0.0211	0.0210
positive HMM	0.3065	0.1575	0.0690	0.0411	0.0374	0.0373	0.0371

It is clear from Table 5.5 that the Kullback-Leibler divergence of a quasi model of a certain order is smaller than the Kullback-Leibler divergence of the positive hidden Markov model of the same order. This observation makes sense. Consider a minimal positive hidden Markov model as well as an equivalent minimal quasi hidden Markov model. As explained before, the order of the quasi hidden Markov model is smaller than or equal to the order of the positive hidden Markov model. On the other hand, when building a quasi hidden Markov model and a positive hidden Markov model of the same order, it is expected that the quasi hidden Markov model performs better than the positive model. It depends on the application which model is best to use. If one needs a physical interpretation of the model, a positive model is needed. However in certain estimation applications, both a positive as a quasi model are fine (see Section 7). In that last case a quasi model is prefered as it gives a better modeling quality for the same order or allows to use a lower order for the same model quality.

It follows from Table 5.5 that the fourth order positive model gives the best trade-off between accuracy and complexity. The obtained model is given by $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ with

$$\begin{array}{rcl}
\Pi(A) & = & \begin{bmatrix} 0.0314 & 0.0075 & 0.0135 & 0.0193 & 0.0007 \\ 0.0221 & 0.0136 & 0.0516 & 0.0684 & 0.0089 \\ 0.0613 & 0.0005 & 0.0668 & 0.0764 & 0.0133 \\ 0.1570 & 0.0737 & 0.1742 & 0.1057 & 0.0006 \\ 0.0493 & 0.0104 & 0.0063 & 0.0427 & 0.0037 \end{bmatrix}, \\
\Pi(C) & = & \begin{bmatrix} 0.0011 & 0.0008 & 0.0059 & 0.0413 & 0.0585 \\ 0.0092 & 0.1469 & 0.0714 & 0.0034 & 0.2340 \\ 0.0157 & 0.0467 & 0.0281 & 0.0824 & 0.0662 \\ 0.0054 & 0.0731 & 0.0082 & 0.0376 & 0.0362 \\ 0.0092 & 0.1207 & 0.0541 & 0.1418 & 0.1044 \end{bmatrix}, \\
\Pi(G) & = & \begin{bmatrix} 0.0487 & 0.1389 & 0.1522 & 0.1252 & 0.0747 \\ 0.0552 & 0.0297 & 0.0974 & 0.0177 & 0.0112 \\ 0.0503 & 0.1292 & 0.1951 & 0.0419 & 0.0560 \\ 0.0423 & 0.0532 & 0.0679 & 0.0182 & 0.0145 \\ 0.0044 & 0.0172 & 0.0288 & 0.0073 & 0.0052 \end{bmatrix}, \\
\Pi(T) & = & \begin{bmatrix} 0.0891 & 0.0532 & 0.0327 & 0.0037 & 0.1015 \\ 0.0683 & 0.0571 & 0.0106 & 0.0002 & 0.0232 \\ 0.0246 & 0.0134 & 0.0045 & 0.0098 & 0.0179 \\ 0.0640 & 0.0353 & 0.0261 & 0.0032 & 0.0035 \\ 0.1457 & 0.0202 & 0.0501 & 0.0082 & 0.1703 \end{bmatrix}, \\
\pi(1) & = & \begin{bmatrix} 0.1937 & 0.2237 & 0.2164 & 0.1703 & 0.1959 \end{bmatrix}.
\end{array}$$

To check the quality of the positive hidden Markov model, we compute the string probabilities generated by $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ for strings of length 4. A selection of these string probabilities (strings starting with A or C) is shown in Table 5.6. By comparing these probabilities to the original probabilities of Table 4.1, we conclude that the modeling works well.

Table 5.6: *We show the string probabilities (for strings starting with A or C) generated by the fifth order hidden Markov model that is obtained by the approximate realization algorithm. By comparing these probabilities to the original string probabilities in Table 4.1, we conclude that the modeling works well.*

	A	C	G	T
AAA	0.0032	0.0026	0.0049	0.0021
CAA	0.0040	0.0034	0.0063	0.0027
ACA	0.0025	0.0021	0.0037	0.0017
CCA	0.0057	0.0046	0.0083	0.0037
AGA	0.0044	0.0034	0.0062	0.0028

CGA	0.0030	0.0023	0.0042	0.0019
ATA	0.0012	0.0009	0.0017	0.0008
CTA	0.0022	0.0016	0.0031	0.0015
AAC	0.0025	0.0037	0.0021	0.0025
CAC	0.0031	0.0045	0.0025	0.0031
ACC	0.0032	0.0051	0.0028	0.0032
CCC	0.0078	0.0127	0.0071	0.0081
AGC	0.0049	0.0080	0.0044	0.0053
CGC	0.0035	0.0054	0.0031	0.0037
ATC	0.0022	0.0034	0.0019	0.0023
CTC	0.0045	0.0063	0.0036	0.0043
AAG	0.0044	0.0060	0.0066	0.0034
CAG	0.0052	0.0070	0.0077	0.0040
ACG	0.0017	0.0023	0.0029	0.0013
CCG	0.0040	0.0056	0.0073	0.0032
AGG	0.0052	0.0072	0.0085	0.0041
CGG	0.0043	0.0059	0.0068	0.0033
ATG	0.0026	0.0033	0.0036	0.0019
CTG	0.0050	0.0063	0.0068	0.0036
AAT	0.0012	0.0025	0.0029	0.0022
CAT	0.0014	0.0031	0.0035	0.0027
ACT	0.0012	0.0027	0.0032	0.0026
CCT	0.0029	0.0064	0.0074	0.0061
AGT	0.0017	0.0037	0.0042	0.0033
CGT	0.0012	0.0026	0.0029	0.0024
ATT	0.0011	0.0026	0.0027	0.0024
CTT	0.0023	0.0053	0.0055	0.0050

5.7 Conclusions

In this chapter we considered the positive realization problem for hidden Markov models. First the current state of the art concerning the exact complete realization problem and the exact partial realization problem is reviewed. Subsequently, the approximate partial realization problem is introduced. It is shown that the approximate Moore realization problem of string probabilities of strings up to length two can be solved using the structured nonnegative matrix factorization. The approximate Mealy partial realization problem of string probabilities of strings up to length t can be solved by generalizing the approach for the Moore realization problem of strings of length two. Simulation examples show that the methods perform well. The methods are applied to the problem of modeling a coin flipping sequence and to the problem of modeling DNA sequences.

Chapter 6

Identification of hidden Markov models

In the identification problem for hidden Markov models, one is given an output string and the problem is to find a hidden Markov model that models the string as well as possible. The classical approach to solve this problem is by use of the Baum-Welch algorithm [10,11,87], an optimization based approach based on maximum likelihood. In [10] results on consistency and asymptotic normality of the maximum likelihood estimator are given, and the conditions for consistency are weakened in [83]. Consistency and asymptotic normality properties of the maximum likelihood estimator are further investigated in [18, 19]. This maximum likelihood approach has very tractable properties. However, the computational complexity of the method is high. In addition, the likelihood surface is multi-modal and the numerical methods may converge only to a local maximum of the objective function.

Identification methods for linear stochastic models on the other hand can be subdivided in two main classes. The first class of identification methods are optimization-based methods such as prediction error methods [75]. In the eighties, subspace identification methods have been introduced in the literature [78]. Subspace identification methods make use of numerically stable operations from linear algebra (SVD, projections, ...). These methods first estimate the state sequence directly from the output data. Subsequently the system matrices are estimated using the obtained state sequence and the given output sequence.

By fitting hidden Markov models in a state space framework [45], an identification procedure can be derived that is analogous to subspace identification of linear stochastic models [7, 8]. However, the models obtained using this method are quasi hidden Markov models. So the procedure of [7, 8] is a *quasi identification* procedure and not a positive identification procedure. To solve several recursive output estimation problems it suffices to have a quasi hidden Markov model (see Chapter 7). However, in other applications a positive hidden Markov model may be needed. It is wrongly stated in [8] that obtaining a positive hidden Markov model from the obtained quasi model is only a matter of finding a correct similarity transformation.

In this chapter we propose a *positive identification* approach. The approach is conceptually related to subspace identification of linear stochastic models. The proposed method first estimates the state sequence directly from the output data and subsequenly estimates the system matrices from the obtained state sequence and the given output sequence.

List of own contributions

We here describe our contributions to the identification problem for hidden Markov models.

- In Section 6.2 we derive the Baum-Welch algorithm for hidden Markov models of Mealy type. To the best of our knowledge, up to this moment, the Baum-Welch formulas have only been considered for Moore hidden Markov models.
- In Section 6.3 we introduce a new approach for the identification of hidden Markov models. The method first estimates the state sequence, and subsequently the system matrices are calculated by solving a least squares problem. In a simulation example we show that the popular Baum-Welch method is outperformed by the proposed method.
- In Section 6.5 we apply the subspace inpired method as well as the Baum-Welch method to the modeling of sequences of the HIV genome. Again the Baum-Welch method is outperformed by the subspace inspired method.

Section-by-section overview

In Section 6.1 we formally state the identification problem. In Section 6.2 we discuss the Mealy Baum-Welch identification procedure for hidden Markov models. In Section 6.3 we introduce a new approach for the identification of hidden Markov inspired by subspace identification for linear stochastic models. In Section 6.4 we perform a simulation example showing that our method outperforms the existing Baum-Welch method. In Section 6.5 we apply the subspace inspired method and the Baum-Welch method to the modeling of HIV sequences. In Section 6.6 finally, we compare the proposed method with the identification of linear stochastic models.

6.1 Introduction

The Mealy identification problem for hidden Markov models can be stated as

Problem 6.1 (Mealy identification problem)**.** *Given an output string* $y_1y_2\ldots y_T$ *of length* T*. Find a Mealy HMM* $(\mathbb{X},\mathbb{Y},\Pi,\pi(1))$ *that models the output string approximately.*

In this chapter we introduce two approaches to the identification of Mealy hidden Markov models. The first approach is based on classical Baum-Welch identification for Moore HMMs. In this approach, the system matrices are estimated directly from the output string. The second approach is inspired by *subspace identification* for linear stochastic systems [78]. In the last approach, the state sequence is estimated directly from the output sequence and subsequently, the system matrices are calculated from the state and output sequence. In Figure 6.1, we schematically show the differences between both approaches.

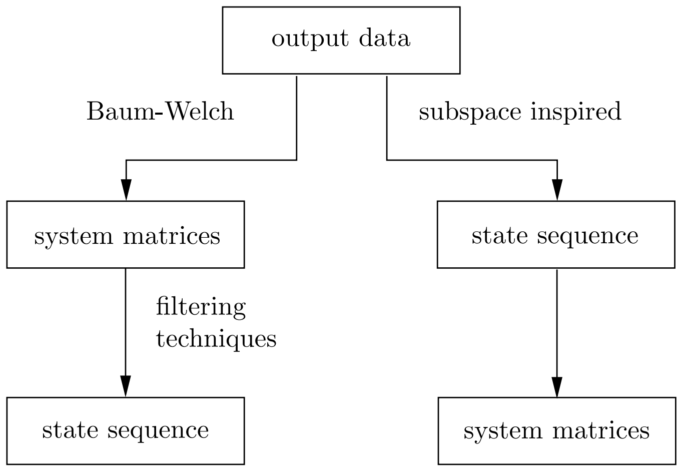


Figure 6.1: *Identification methods aim at constructing models from output data. The left hand side shows the classical approach: first obtain the system matrices, then estimate the state sequence if needed. The right hand side shows the approach inspired by subspace identification for linear stochastic models: first the states are estimated directly from data, then the system matrices can be obtained.*

6.2 Baum-Welch identification

In this section we extend the Baum-Welch algorithm to the identification of Mealy HMMs. To the best of our knowledge, in the literature, Baum-Welch has only been considered for Moore HMMs. Baum-Welch is a Maximum Likelihood (ML) method which means that the system matrices $\Pi(\mathrm{y}), \mathrm{y} \in \mathbb{Y}$ and $\pi(1)$ are estimated such that the likelihood of the observed string is maximized, where the likelihood is defined as $P(y(1,2,\ldots,T) = y_1 y_2 \ldots y_T \mid \lambda)$ where

λ denotes the set of model parameters. The maximum likelihood problem is solved using Expectation-Maximization, i.e. starting with an initial guess for the model parameters λ and updating them iteratively such that the likelihood is nondecreasing in each step. We review the Expectation-Maximization approach in Appendix C.

We now apply the Expectation-Maximization approach to the identification of hidden Markov models. Consider $\mathbf{y} = y_1 y_2 \ldots y_T$ to be the observed data and the underlying state sequence $\mathbf{x} = x_1 x_2 \ldots x_{T+1}$ to be hidden or unobserved. The model parameters are given by $\lambda = (\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$. The function $Q(\lambda, \lambda^{(k)})$ (Equation (C.3)) now becomes

$$Q(\lambda, \lambda^{(k)}) = \sum_{\mathbf{x} \in \mathbb{X}^{T+1}} P(\mathbf{x}, \mathbf{y} | \lambda^{(k)}) \log P(\mathbf{x}, \mathbf{y} | \lambda)$$

where $\lambda^{(k)} = (\mathbb{X}, \mathbb{Y}, \Pi^{(k)}, \pi^{(k)}(1))$ is the previous estimate of the parameters. Given a particular state sequence $\mathbf{x}$, representing $P(\mathbf{x}, \mathbf{y}|\lambda)$ is quite easy, i.e.

$$P(\mathbf{x}, \mathbf{y}|\lambda) = \pi_{x_1}(1) \prod_{t=1}^{T} \Pi_{x_t, x_{t+1}}(y_t).$$

The function $Q(\lambda, \lambda^{(k)})$ then becomes

$$\begin{aligned} Q(\lambda, \lambda^{(k)}) &= \sum_{\mathbf{x} \in \mathbb{X}^{T+1}} \log \pi_{x_1}(1) P(\mathbf{y}, \mathbf{x}|\lambda^{(k)}) \\ &\quad + \sum_{\mathbf{x} \in \mathbb{X}^{T+1}} \left(\sum_{t=1}^{T} \log \Pi_{x_t, x_{t+1}}(y_t) \right) P(\mathbf{y}, \mathbf{x}|\lambda^{(k)}). \quad (6.1) \end{aligned}$$

Since the parameters we wish to optimize are now independently split into the two terms in the sum, we can optimize each term individually. One can easily see that the first term in (6.1) can be rewritten as

$$\sum_{\mathbf{x} \in \mathbb{X}^{T+1}} \log \pi_{x_1}(1) P(\mathbf{y}, \mathbf{x}|\lambda^{(k)}) = \sum_{i=1}^{|\mathbb{X}|} \log \pi_i(1) P(\mathbf{y}, x_1 = i|\lambda^{(k)}).$$

Adding the Lagrange multiplier γ, using the constraint that $\sum_i \pi_i(1) = 1$, and setting the derivative equal to zero, we get

$$\frac{\partial}{\partial \pi_i} \left(\sum_{i=1}^{|\mathbb{X}|} \log \pi_i(1) P(\mathbf{y}, x_1 = i|\lambda^{(k)}) + \gamma (\sum_{i=1}^{|\mathbb{X}|} \pi_i(1) - 1) \right) = 0.$$

We get

$$\pi_i^{(k+1)}(1) = \frac{P(\mathbf{y}, x_1 = i|\lambda^{(k)})}{P(\mathbf{y}|\lambda^{(k)})}. \quad (6.2)$$

The second term in (6.1) can be rewritten as

$$\sum_{\mathbf{x}\in\mathbb{X}^{T+1}}\left(\sum_{t=1}^{T}\log\Pi_{x_t,x_{t+1}}(y_t)\right)P(\mathbf{y},\mathbf{x}|\lambda^{(k)}) =$$
$$\sum_{i=1}^{|\mathbb{X}|}\sum_{j=1}^{|\mathbb{X}|}\left(\sum_{t=1}^{T}\log\Pi_{ij}(y_t)\right)P(\mathbf{y},x_t=i,x_{t+1}=j|\lambda^{(k)}).$$

In a similar way, we can use Lagrange multipliers with the constraints $\sum_{j=1}^{|\mathbb{X}|}\sum_{\mathrm{y}\in\mathbb{Y}}\Pi_{ij}(\mathrm{y})=1, i=1,\ldots,|\mathbb{X}|$ to get

$$\Pi_{ij}^{(k+1)}(\mathrm{y}) = \frac{\sum_{t=1}^{T}P(\mathbf{y},x_t=i,x_{t+1}=j|\lambda^{(k)})\delta(y_t,\mathrm{y})}{\sum_{t=1}^{T}P(\mathbf{y},x_t=i|\lambda^{(k)})}. \tag{6.3}$$

Define the variables $\gamma_i^{(k)}(t)$ and $\xi_{ij}^{(k)}(t)$ as

$$\begin{aligned}\gamma_i^{(k)}(t) &:= P(x(t)=i|y(1,\ldots,T)=y_1\ldots y_T,\lambda^{(k)}),\\ \xi_{ij}^{(k)}(t) &:= P(x(t)=i,x(t+1)=j|y(1,\ldots,T)=y_1\ldots y_T,\lambda^{(k)}),\end{aligned}$$

then (6.2) and (6.3) become

$$\begin{aligned}\pi_i^{(k+1)}(1) &= \gamma_i^{(k)}(1),\\ \Pi_{ij}^{(k+1)}(\mathrm{y}) &= \frac{\sum_{t=1}^{T}\delta(y_t,\mathrm{y})\xi_{ij}^{(k)}(t)}{\sum_{t=1}^{T}\gamma_i^{(k)}(t)}.\end{aligned}$$

The variables $\gamma_i^{(k)}(t)$ and $\xi_{ij}^{(k)}(t)$ can be calculated as

$$\begin{aligned}\gamma_i^{(k)}(t) &= \frac{\alpha_i^{(k)}(t-1)\beta_i^{(k)}(t)}{\alpha^{(k)}(t-1)\beta^{(k)}(t)},\\ \xi_{ij}^{(k)}(t) &= \frac{\alpha_i^{(k)}(t-1)\beta_j^{(k)}(t+1)\Pi_{ij}^{(k)}(y_t)}{\alpha^{(k)}(t-1)\beta^{(k)}(t)},\end{aligned}$$

where $\alpha^{(k)}(t)\in\mathbb{R}^{1\times|\mathbb{X}|}$ are the *forward variables* and $\beta^{(k)}(t)\in\mathbb{R}^{|\mathbb{X}|\times 1}$ the *backward variables* defined as

$$\begin{aligned}\alpha_i^{(k)}(t) &:= P(x(t+1)=i,y(1,\ldots,t)=y_1\ldots y_t|\lambda^{(k)})\\ \beta_i^{(k)}(t) &:= P(y(t,\ldots,T)=y_t\ldots y_T \mid x(t)=i,\lambda^{(k)}).\end{aligned}$$

The forward variables can be calculated inductively as

$$\begin{aligned}\alpha^{(k)}(0) &= \pi^{(k)},\\ \alpha^{(k)}(t+1) &= \alpha^{(k)}(t)\Pi^{(k)}(y_{t+1}),\end{aligned}$$

while the backward variables can be calculated as

$$\begin{aligned}\beta^{(k)}(T+1) &= e,\\ \beta^{(k)}(t) &= \Pi^{(k)}(y_t)\beta^{(k)}(t+1).\end{aligned}$$

Below we summarize the Baum-Welch algorithm for Mealy HMMs.

Algorithm 6.1. *Given an output sequence* $\mathbf{y} = y_1 y_2 \dots y_T$*, and an initial guess of the model* $\lambda^{(0)} = (\mathbb{X}, \mathbb{Y}, \Pi^{(0)}, \pi^{(0)}(1))$ *with state dimension* $|\mathbb{X}|$*. Iterate the following steps for* $k = 0, 1, 2, \dots$ *until convergence:*

1. *Compute the forward and backward variables for the given output sequence* $\mathbf{y}$ *and model* $\lambda^{(k)}$*.*

2. *Calculate the variables* $\gamma_i^{(k)}(t)$ *and* $\xi_{ij}^{(k)}(t)$ *from the forward and the backward variables.*

3. *Obtain an updated model* $\lambda^{(k+1)} = (\mathbb{X}, \mathbb{Y}, \Pi^{(k+1)}, \pi^{(k+1)}(1))$ *using*

$$\begin{aligned} \pi_i^{(k+1)}(1) &= \gamma_i^{(k)}(1), \\ \Pi_{ij}^{(k+1)}(\boldsymbol{y}) &= \frac{\sum_{t=1}^{T-1} \delta(y_t, \boldsymbol{y}) \xi_{ij}^{(k)}(t)}{\sum_{t=1}^{T-1} \gamma_i^{(k)}(t)}. \end{aligned}$$

6.3 Subspace inspired identification

In this section we explain a new approach to the identification problem of stationary hidden Markov models. The approach consists of two steps. In the first step (Section 6.3.2) the underlying state process is estimated directly from the given output string. In the second step (Section 6.3.3) the system matrices are calculated from the obtained state sequence and the given output sequence. We start with introducing some notation (Section 6.3.1).

6.3.1 Notation

System related matrices Consider a Mealy HMM $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$, then for given $i_1 \in \mathbb{Z}_+$ and $i_2 \in \mathbb{Z}_+$ the matrix $\mathfrak{H}_{(i_1+1, i_2+1)}$ is defined as

$$\left(\mathfrak{H}_{(i_1+1,i_2+1)}\right)_{kl} := \mathcal{P}(\mathbf{u}_k \mathbf{v}_l),$$

where $\mathbf{u}_k$ is the k-th element of the set of strings from $\mathbb{Y}^{i_1}$ in first lexicographical ordering $\mathcal{U}_{(i_1)}$ and $\mathbf{v}_l$ is the l-th element of the set of strings from $\mathbb{Y}^{i_2}$ in last lexicographical ordering $\mathcal{V}_{(i_2)}$. The matrix $\mathfrak{H}_{(i_1+1,i_2+1)}$ can be decomposed as

$$\mathfrak{H}_{(i_1+1,i_2+1)} = VH, \tag{6.4}$$

with

$$\begin{aligned} V &= \begin{bmatrix} \pi(1)\Pi(\mathbf{u}_1) \\ \pi(1)\Pi(\mathbf{u}_2) \\ \vdots \\ \pi(1)\Pi(\mathbf{u}_{|\mathbb{Y}|^{i_1}}) \end{bmatrix}, \\ H &= \begin{bmatrix} \Pi(\mathbf{v}_1)e & \Pi(\mathbf{v}_2)e & \dots & \Pi(\mathbf{v}_{|\mathbb{Y}|^{i_2}})e \end{bmatrix}, \end{aligned}$$

where $V \in \mathbb{R}_+^{|\mathbb{Y}|^{i_1} \times |\mathbb{X}|}$ and $H \in \mathbb{R}_+^{|\mathbb{X}| \times |\mathbb{Y}|^{i_2}}$. The elements of V are equal to

$$V_{ki} = P(y(1, 2, \ldots, i_1) = \mathbf{u}_k, x(i_1 + 1) = i),$$

while the elements of H are equal to

$$H_{il} = P(y(i_1 + 1, i_1 + 2, \ldots, i_1 + i_2) = \mathbf{v}_l | x(i_1 + 1) = i).$$

The matrix $\mathfrak{H}_{(i_1+2,i_2+1)}$ on the other hand, can be decomposed as

$$\mathfrak{H}_{(i_1+2,i_2+1)} = WH, \tag{6.5}$$

with H as before and

$$W = \begin{bmatrix} \pi(1)\Pi(\mathrm{y}_1\mathbf{u}_1) \\ \vdots \\ \pi(1)\Pi(\mathrm{y}_{|\mathbb{Y}|}\mathbf{u}_1) \\ \pi(1)\Pi(\mathrm{y}_1\mathbf{u}_2) \\ \vdots \\ \pi(1)\Pi(\mathrm{y}_{|\mathbb{Y}|}\mathbf{u}_{|\mathbb{Y}|^{i_1}}) \end{bmatrix}.$$

State distribution matrices The *state distribution matrix* $\tilde{X}_{i_1} \in [0, 1]^{(T-i_1)\times|\mathbb{X}|}$ is defined as

$$\tilde{X}_{i_1} := \begin{bmatrix} \tilde{x}(i_1 + 1) \\ \tilde{x}(i_1 + 2) \\ \vdots \\ \tilde{x}(T) \end{bmatrix},$$

where $\tilde{x}(t)$ is a row vector in $\mathbb{R}^{1\times|\mathbb{X}|}$ defined as

$$\tilde{x}_i(t) := P(x(t) = i | y(t - i_1, ..., t - 1) = y_{t-i_1} ... y_{t-1}).$$

The *next-state distribution matrix* $\tilde{X}^+_{i_1+1} \in [0, 1]^{(T-i_1)\times|\mathbb{X}|}$ is defined as

$$\tilde{X}^+_{i_1+1} = \begin{bmatrix} \tilde{x}^+(i_1 + 2) \\ \tilde{x}^+(i_1 + 3) \\ \vdots \\ \tilde{x}^+(T + 1) \end{bmatrix},$$

where $\tilde{x}^+(t + 1)$ is a row vector in $\mathbb{R}^{1\times|\mathbb{X}|}$ defined as

$$\tilde{x}^+_i(t + 1) := P(x(t + 1) = i | y(t - i_1, ..., t - 1) = y_{t-i_1} ... y_{t-1}).$$

The *next-state-output distribution matrix* $\tilde{X}^{\mathrm{y}}_{i_1+1} \in [0, 1]^{(T-i_1)\times|\mathbb{X}|}$ is defined as

$$\tilde{X}^{\mathrm{y}}_{i_1+1} = \begin{bmatrix} \tilde{x}^{\mathrm{y}}(i_1 + 2) \\ \tilde{x}^{\mathrm{y}}(i_1 + 3) \\ \vdots \\ \tilde{x}^{\mathrm{y}}(T + 1) \end{bmatrix},$$

where $\tilde{x}^{\mathtt{y}}(t+1)$ is a row vector in $\mathbb{R}^{1\times|\mathbb{X}|}$ defined as

$$\tilde{x}_i^{\mathtt{y}}(t+1) := P(x(t+1)=i, y(t)=\mathtt{y}|y(t-i_1,...,t-1)=y_{t-i_1}...y_{t-1}).$$

The next-state-output distribution matrix can be calculated from the next-state distribution matrix using

$$\tilde{x}^{\mathtt{y}}(t+1) = \begin{cases} \tilde{x}^{+}(t+1) & \text{if } y_t = \mathtt{y}, \\ \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix} & \text{else.} \end{cases}$$

6.3.2 Estimating the state distribution matrices

In this section we explain how the state distribution matrix and the next-state distribution matrix can be estimated directly from output data. In a first step we describe a method to find the state distribution matrix and the next-state distribution matrix for a given output sequence $y_1y_2\dots y_T$ under the assumption that the matrices $\mathfrak{H}_{(i_1+1,i_2+1)}$ and $\mathfrak{H}_{(i_1+2,i_2+1)}$ of the underlying HMM are given for certain choices of i_1 and i_2 as well as the nonnegative decompositions $\mathfrak{H}_{(i_1+1,i_2+1)} = VH$ and $\mathfrak{H}_{(i_1+2,i_2+1)} = WH$. In a second step we explain how the matrices $\mathfrak{H}_{(i_1+1,i_2+1)}$ and $\mathfrak{H}_{(i_1+2,i_2+1)}$ can be estimated from output data. Moreover, we show that the nonnegative decomposition of these matrices can be estimated using approximate nonnegative matrix factorization techniques of Section 2.2. By combining both steps we have a method to find the estimated state distribution matrices directly from output data.

Estimating the state distribution matrices given $\mathfrak{H}_{(i_1+1,i_2+1)} = VH$ and $\mathfrak{H}_{(i_1+2,i_2+1)} = WH$

Define $\tilde{V}$ as

$$\tilde{V} := (\operatorname{diag}(Ve_{|\mathbb{X}|}))^{-1}V,$$

with V defined as in Section 6.3.1. Note that it holds, due to stationarity, that

$$\tilde{V}_{ki} = P(x(t)=i|y(t-i_1,\dots,t-1)=\mathbf{u}_k),$$

where $\mathbf{u}_k$ is the k-th element of the set of strings from $\mathbb{Y}^{i_1}$ in first lexicographical ordering $\mathcal{U}_{(i_1)}$. Now the state distribution matrix $\tilde{X}_{i_1}$ can be calculated using

$$\tilde{x}(t) = \tilde{V}_{\kappa,:},$$

with κ the position of the string $y_{t-i_1}\dots y_{t-1}$ in the first lexicographical ordering $\mathcal{U}_{(i_1)}$ of the strings of length i_1. Note that the state distribution matrix $\tilde{X}_{i_1}$ is obtained by running a bank of state predictors (Algorithm 7.1). Figure 6.2 illustrates this concept. The bank of state predictors runs in horizontal direction (over the rows). It should be noted that the state estimators only use partial output information. For instance the q-th row of $\tilde{X}_{i_1}$ is equal to $\tilde{x}(i_1+q)$ and is based on the measurements $y_qy_{q+1}\dots y_{i_1+q-1}$ instead of all output measurements up until time i_1+q-1 (as would be expected).

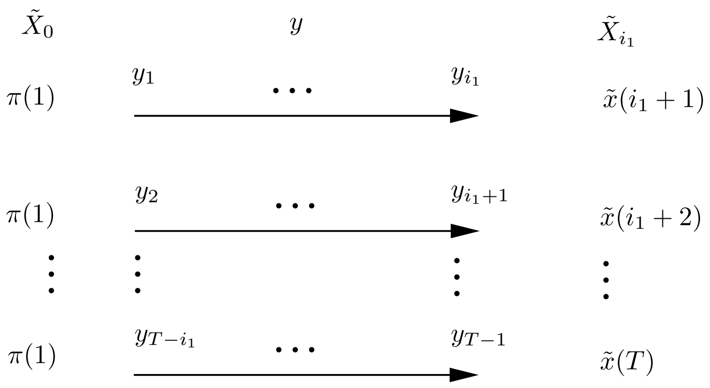


Figure 6.2: *The state distribution matrix $\tilde{X}_{i_1}$ is obtained by running a bank of state predictors. The bank of state predictors runs in horizontal direction (over the rows). The state estimators use only partial output information. For instance the q-th row of $\tilde{X}_{i_1}$ is equal to $\tilde{x}(i_1+q)$ and is based on the measurements $y_q y_{q+1} \cdots y_{i_1+q-1}$.*

The next-state distribution matrix can be calculated from W using

$$\begin{aligned}
\bar{W} &= \begin{bmatrix} e_{|\mathbb{Y}|}^\top & & & \\ & e_{|\mathbb{Y}|}^\top & & \\ & & \ddots & \\ & & & e_{|\mathbb{Y}|}^\top \end{bmatrix} W, \\
\tilde{W} &= (\operatorname{diag}(\bar{W} e_{|\mathbb{X}|}))^{-1} \bar{W}, \\
\tilde{x}^+(t+1) &= \tilde{W}_{\kappa,:}
\end{aligned}$$

where κ is the position of the string $y_{t-i_1} \dots y_{t-1}$ in the first lexicographical ordering of the strings of length i_1.

We now state that the predicted state distributions stacked in the state distribution matrix converge to the true predicted state distributions (calculated using Algorithm 7.1) as i_1 goes to infinity.

Theorem 6.1. *The probabilities $\tilde{x}(t) = P(x(t)|y(t-i_1, ..., t-1) = y_{t-i_1}...y_{t-1})$ stacked in the state distribution matrix $\tilde{X}_{i_1}$ converge to the predicted state distributions $\pi(t|y_1, y_2, ..., y_{t-1}) = P(x(t)|y(1, 2, ..., t-1) = y_1 y_2 ... y_{t-1})$ (calculated using Algorithm 7.1) as i_1 goes to infinity.*

An analogous property can be given for the predicted next-state distributions stacked in the next-state distribution matrix.

Estimating $\mathfrak{H}_{(i_1+1,i_2+1)}$ and $\mathfrak{H}_{(i_1+2,i_2+1)}$ and their nonnegative decomposition from data

So far we described a method to estimate the state distribution matrix and the next-state distribution matrix corresponding to the given output sequence. The method supposes that for certain choices of i_1 and i_2 the matrices $\mathfrak{H}_{(i_1+1,i_2+1)}$ and $\mathfrak{H}_{(i_1+2,i_2+1)}$ containing string probabilities are given. Moreover it is supposed that the nonnegative decompositions (6.4) and (6.5) of these matrices are given. We now show that the matrices $\mathfrak{H}_{(i_1+1,i_2+1)}$ and $\mathfrak{H}_{(i_1+2,i_2+1)}$ can be estimated from data and that the nonnegative decomposition of these matrices can be estimated using the approximate nonnegative matrix factorization technique. As a result the complete procedure to find the state distribution matrices works directly from the given output data.

The matrices $\mathfrak{H}_{(i_1+1,i_2+1)}$ and $\mathfrak{H}_{(i_1+2,i_2+1)}$ contain string probabilities of strings of length i_1+i_2 and i_1+i_2+1. It is possible to estimate these string probabilities directly from the output sequence (using Equation (3.3)). The estimated matrices are denoted by $\tilde{\mathfrak{H}}_{(i_1+1,i_2+1)}$ and $\tilde{\mathfrak{H}}_{(i_1+2,i_2+1)}$.

The nonnegative decomposition of the matrices $\tilde{\mathfrak{H}}_{(i_1+1,i_2+1)}$ and $\tilde{\mathfrak{H}}_{(i_1+2,i_2+1)}$ can be obtained by applying the nonnegative matrix factorization (Theorem 2.6) to find an approximate decomposition of the form

$$\begin{bmatrix} \tilde{\mathfrak{H}}_{(i_1+1,i_2+1)} \\ \tilde{\mathfrak{H}}_{(i_1+2,i_2+1)} \end{bmatrix} \simeq \begin{bmatrix} V^{\text{est}} \\ W^{\text{est}} \end{bmatrix} H^{\text{est}}.$$

As there does not exist a practical useful procedure to determine the minimum inner dimension for which such a decomposition exists, we need to chose the inner dimension. Notice that the choice of the inner dimension is important as it is the state dimension of the obtained model. So, as usual in system identification, the proposed identification method provides a model for a user-defined model order.

6.3.3 Calculating the system matrices

In this section we explain how the system matrices can be obtained from the state distribution matrix and the next-state-output distribution matrix.

It holds for $t = i_1+1, \ldots, T$ and for $\mathtt{y} \in \mathbb{Y}$ that

$$\tilde{x}^{\mathtt{y}}(t+1) \quad = \quad \tilde{x}(t)\Pi(\mathtt{y}). \tag{6.6}$$

Equation (6.6) can be rewritten as

$$\begin{bmatrix} \tilde{X}^{\mathtt{y}_1}_{i_1+1} & \tilde{X}^{\mathtt{y}_2}_{i_1+1} & \ldots & \tilde{X}^{\mathtt{y}_{|\mathbb{Y}|}}_{i_1+1} \end{bmatrix} = \tilde{X}_{i_1} \begin{bmatrix} \Pi(\mathtt{y}_1) & \Pi(\mathtt{y}_2) & \ldots & \Pi(\mathtt{y}_{|\mathbb{Y}|}) \end{bmatrix}. \tag{6.7}$$

Solving Equation (6.7) for $\begin{bmatrix} \Pi(\mathtt{y}_1) & \Pi(\mathtt{y}_2) & \ldots & \Pi(\mathtt{y}_{|\mathbb{Y}|}) \end{bmatrix}$ does not necessarily yield nonnegative estimates for $\Pi(\mathtt{y}), \mathtt{y} \in \mathbb{Y}$ and hence does not give a solution to the identification problem. As will be shown in Theorem 6.2, the solution to the identification problem lies in the use of the state sequence matrix, the next-state sequence matrix and the next-state-output sequence matrix, defined below.

The *state sequence matrix* $\hat{X}_{i_1} \in \{0,1\}^{(T-i_1)\times|\mathbb{X}|}$ is defined as

$$\hat{X}_{i_1} = \begin{bmatrix} \hat{x}(i_1+1) \\ \hat{x}(i_1+2) \\ \vdots \\ \hat{x}(T) \end{bmatrix},$$

where $\hat{x}(t) \in \mathbb{R}^{1\times|\mathbb{X}|}$ is defined as

$$\hat{x}_i(t) = \begin{cases} 1 & \text{if } i = \text{ML}(x(t)|y(t-i_1,...,t-1) = y_{t-i_1}...y_{t-1}), \\ 0 & \text{else.} \end{cases}$$

Notice that $\hat{x}(t)$ is a row vector of size $|\mathbb{X}|$ with all elements equal to zero except for the element at position i which is equal to 1, where i is the most likely estimate for $x(t)$ given the past i_1 observations.

The *next-state sequence matrix* $\hat{X}^+_{i_1+1} \in [0,1]^{(T-i_1)\times|\mathbb{X}|}$ is defined as

$$\hat{X}^+_{i_1+1} = \begin{bmatrix} \hat{x}^+(i_1+2) \\ \hat{x}^+(i_1+3) \\ \vdots \\ \hat{x}^+(T+1) \end{bmatrix},$$

where $\hat{x}^+(t+1) \in \mathbb{R}^{1\times|\mathbb{X}|}$ is equal to

$$\hat{x}^+_i(t+1) = \begin{cases} 1 & \text{if } i = \text{ML}(x(t+1)|y(t-i_1,...,t-1) = y_{t-i_1}...y_{t-1}), \\ 0 & \text{else.} \end{cases}$$

The *next-state-output sequence matrix* $\hat{X}^{\mathtt{y}}_{i_1+1} \in \{0,1\}^{(T-i_1)\times|\mathbb{X}|}$ is defined as

$$\hat{X}^{\mathtt{y}}_{i_1+1} = \begin{bmatrix} \hat{x}^{\mathtt{y}}(i_1+2) \\ \hat{x}^{\mathtt{y}}(i_1+3) \\ \vdots \\ \hat{x}^{\mathtt{y}}(T+1) \end{bmatrix},$$

where $\hat{x}^{\mathtt{y}}(t+1) \in \mathbb{R}^{1\times|\mathbb{X}|}$ is equal to

$$\hat{x}^{\mathtt{y}}_i(t{+}1) = \begin{cases} 1 & \text{if } i = \text{ML}(x(t+1), y(t) = \mathtt{y}|y(t-i_1,...,t-1) = y_{t-i_1}...y_{t-1}) \\ & \text{and } y(t) = \mathtt{y} \\ 0 & \text{else.} \end{cases}$$

Notice that the state sequence matrix and the next-state sequence matrix can be calculated from the state distribution matrix and the next-state distribution matrix using

$$\hat{x}_i(t) \;\; = \;\; \begin{cases} 1 & i = \text{argmax}_k\, \tilde{x}_k(t), \\ 0 & \text{else,} \end{cases}$$

$$\hat{x}_i^+(t) \quad = \quad \begin{cases} 1 & i = \text{argmax}_k \, \tilde{x}_k^+(t), \\ 0 & \text{else.} \end{cases}$$

The next-state-output sequence matrix can be calculated from the next-state sequence matrix using

$$\hat{x}^{\mathrm{y}}(t+1) = \begin{cases} \hat{x}^+(t+1) & \text{if } y_t = \mathrm{y}, \\ \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix} & \text{else.} \end{cases}$$

Now the "most likely version" of Equation (6.7) becomes

$$\begin{bmatrix} \hat{X}_{i_1+1}^{\mathrm{y}_1} & \hat{X}_{i_1+1}^{\mathrm{y}_2} & \dots & \hat{X}_{i_1+1}^{\mathrm{y}_{|\mathbb{Y}|}} \end{bmatrix} = \hat{X}_{i_1} \begin{bmatrix} \Pi(\mathrm{y}_1) & \Pi(\mathrm{y}_2) & \dots & \Pi(\mathrm{y}_{|\mathbb{Y}|}) \end{bmatrix}. \quad (6.8)$$

By solving (6.8) for $\Pi(\mathrm{y}), \ \mathrm{y} \in \mathbb{Y}$ in least squares sense, we find

$$\begin{bmatrix} \Pi(\mathrm{y}_1) & \Pi(\mathrm{y}_2) & \dots & \Pi(\mathrm{y}_{|\mathbb{Y}|}) \end{bmatrix} = (\hat{X}_{i_1})^\dagger \begin{bmatrix} \hat{X}_{i_1+1}^{\mathrm{y}_1} & \hat{X}_{i_1+1}^{\mathrm{y}_2} & \dots & \hat{X}_{i_1+1}^{\mathrm{y}_{|\mathbb{Y}|}} \end{bmatrix} \quad (6.9)$$

where $(\hat{X}_{i_1})^\dagger = (\text{diag}(e^\top \hat{X}_{i_1}))^{-1}(\hat{X}_{i_1})^\top$ is the Moore-Penrose pseudo-inverse of $\hat{X}_{i_1}$. We are now able to prove the following theorem.

Theorem 6.2. *The matrices $\Pi(y_1), \Pi(y_2), \dots, \Pi(y_{|\mathbb{Y}|})$ calculated using Equation (6.9) are elementwise nonnegative and fullfill*

$$\left(\sum_y \Pi(y) \right) e = e.$$

Proof: One can easily see that $(\hat{X}_{i_1})^\dagger = (\text{diag}(e^\top \hat{X}_{i_1}))^{-1}(\hat{X}_{i_1})^\top$ is elementwise nonnegative. Hence, $\begin{bmatrix} \Pi(\mathrm{y}_1) & \Pi(\mathrm{y}_2) & \dots & \Pi(\mathrm{y}_{|\mathbb{Y}|}) \end{bmatrix}$ is the product of two nonnegative matrices and is itself nonnegative. From Equation (6.9) it follows that

$$\sum_{\mathrm{y}} \Pi(\mathrm{y}) = (\hat{X}_{i_1})^\dagger \hat{X}_{i_1+1}^+.$$

Multiplying both sides with e at the right hand side yields after some calculation

$$\left(\sum_{\mathrm{y}} \Pi(\mathrm{y}) \right) e = (\hat{X}_{i_1})^\dagger e = e.$$

■

The initial state distribution $\pi(1)$ can be calculated as the normalised left eigenvector of $\sum_{\mathrm{y}} \Pi(\mathrm{y})$ corresponding to the eigenvalue 1.

We end this section with some comments on the choice of the parameters i_1 and i_2. On the one hand the parameters i_1 and i_2 need to be large such that the positive rank of the matrices $\mathfrak{H}_{(i_1+1,i_2+1)}$ and $\mathfrak{H}_{(i_1+1,i_2+2)}$ is larger than or equal to the desired system order $|\mathbb{X}|$ and such that the state estimates in the state sequence matrix are as good as possible. On the other hand, the larger i_1 and i_2, the smaller the accuracy of the estimates of the string probabilities stacked in the matrices $\mathfrak{H}_{(i_1+1,i_2+1)}$ and $\mathfrak{H}_{(i_1+1,i_2+2)}$. Also, the larger i_1 and i_2, the higher the computational complexity to compute the matrices $\mathfrak{H}_{(i_1+1,i_2+1)}$ and $\mathfrak{H}_{(i_1+1,i_2+1)}$ and to compute the state distribution matrix. As a rule of thumb, we take $i_1 \simeq i_2 \simeq |\mathbb{X}|$.

6.3.4 Summary of the algorithm

In this section we summarize the identification algorithm that was described in the previous sections.

Algorithm 6.1. *Given an output sequence* $y_1, y_2, \ldots y_T$*, a state set* $\mathbb{X} = \{1, 2, \ldots, |\mathbb{X}|\}$*, and integers* i_1 *and* i_2*. Perform the following steps to find a Mealy model* $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ *of the output sequence.*

1. *Estimate the matrices* $\mathfrak{H}_{(i_1+1, i_2+1)}$ *and* $\mathfrak{H}_{(i_1+2, i_2+1)}$ *from the output data using Equation (3.3).*
2. *Calculate the simultaneous nonnegative decomposition using the nonnegative matrix factorization*
$$\begin{aligned} \mathfrak{H}_{(i_1+1, i_2+1)} &= VH, \\ \mathfrak{H}_{(i_1+2, i_2+1)} &= WH. \end{aligned}$$
3. *Calculate the state distribution matrix* $\tilde{X}_{i_1}$ *and the next-state-output distribution matrices* $\tilde{X}^{y}_{i_1}$ *(Section 6.3.2).*
4. *Calculate the state sequence matrix* $\hat{X}_{i_1}$ *and the next-state-output sequence matrices* $\hat{X}^{y}_{i_1}$ *(Section 6.3.3).*
5. *Calculate the matrices* $\Pi(y), y \in \mathbb{Y}$ *as*
$$\left[\begin{array}{cccc} \Pi(y_1) & \Pi(y_2) & \ldots & \Pi(y_{|\mathbb{Y}|}) \end{array}\right] = (\hat{X}_{i_1})^{\dagger} \left[\begin{array}{cccc} \hat{X}^{y_1}_{i_1+1} & \hat{X}^{y_2}_{i_1+1} & \ldots & \hat{X}^{y_{|\mathbb{Y}|}}_{i_1+1} \end{array}\right].$$
6. *Calculate the initial state distribution* $\pi(1)$ *as the normalised left eigenvector of* $\sum_y \Pi(y)$ *corresponding to the eigenvalue 1.*

6.4 Simulation example

In this simulation example we are given an output string $y_1 \ldots y_{1000}$ generated by $\lambda_{\text{true}} = (\{1, 2\}, \{1, 2\}, \Pi_{\text{true}}, \pi_{\text{true}}(1))$ where

$$\begin{aligned} \Pi_{\text{true}}(1) &= \left[\begin{array}{cc} 0.20 & 0.40 \\ 0.00 & 0.20 \end{array}\right], \\ \Pi_{\text{true}}(2) &= \left[\begin{array}{cc} 0.10 & 0.30 \\ 0.80 & 0.00 \end{array}\right], \\ \pi_{\text{true}}(1) &= \left[\begin{array}{cc} 0.53 & 0.47 \end{array}\right]. \end{aligned}$$

In fact this model is unknown, but we give it here the check the performance of our algorithm. We now use the modified Baum-Welch algorithm as well as the method that first estimates the state sequence and then the system matrices to find a Mealy hidden Markov model of order $|\mathbb{X}| = 2$.

The model found with Baum-Welch (after convergence) is given by $\lambda_{\text{BW}} = (\{1,2\},\{1,2\},\Pi_{\text{BW}},\pi_{\text{BW}}(1))$ where

$$\begin{aligned}
\Pi_{\text{BW}}(1) &= \begin{bmatrix} 0.0736 & 0.0986 \\ 0.5311 & 0.1415 \end{bmatrix}, \\
\Pi_{\text{BW}}(2) &= \begin{bmatrix} 0.0751 & 0.7526 \\ 0.2424 & 0.0850 \end{bmatrix}, \\
\pi_{\text{BW}}(1) &= \begin{bmatrix} 0 & 1 \end{bmatrix},
\end{aligned}$$

while the model found with the subspace inspired method with $i_1 = i_2 = 3$ is given by $\lambda_{\text{SS}} = (\{1,2\},\{1,2\},\Pi_{\text{SS}},\pi_{\text{SS}}(1))$ with

$$\begin{aligned}
\Pi_{\text{SS}}(1) &= \begin{bmatrix} 0.0699 & 0.2574 \\ 0.5651 & 0.0000 \end{bmatrix}, \\
\Pi_{\text{SS}}(2) &= \begin{bmatrix} 0.1342 & 0.5386 \\ 0.4349 & 0.0000 \end{bmatrix}, \\
\pi_{\text{SS}}(1) &= \begin{bmatrix} 0.5568 & 0.4432 \end{bmatrix}.
\end{aligned}$$

The system matrices of a Mealy HMM are not uniquely determined (Section 3.3.1.2), so it is not possible to decide which model is best by comparing the system matrices. The check the quality of both estimated models, we define a divergence measure between the estimated model and the true model. A popular divergence measure between λ_{true} and its approximation λ_{approx} is the Kullback-Leibler divergence defined as

$$D_{\text{KL}}(\lambda_{\text{true}}||\lambda_{\text{approx}}) := \sum_{\mathbf{y}\in\mathbb{Y}^*} \mathcal{P}(\mathbf{y}|\lambda_{\text{true}}) \log \frac{\mathcal{P}(\mathbf{y}|\lambda_{\text{true}})}{\mathcal{P}(\mathbf{y}|\lambda_{\text{approx}})},$$

where $\mathcal{P}(\text{y}|\lambda)$ denotes the probability of the string y for the model λ. To be able to calculate this distance in practice, we take only strings up to a certain length t instead of all strings of finite length. As strings probabilities of strings of length smaller than t can be calculated from strings probabilities of strings of length t, we do not take them into account for the calculation of the divergence between the models. As a practical divergence between two hidden Markov models, we use

$$D^{(t)}_{\text{KL}}(\lambda_{\text{true}}||\lambda_{\text{approx}}) = \sum_{\mathbf{y}\in\mathbb{Y}^t} \mathcal{P}(\mathbf{y}|\lambda_{\text{true}}) \log \frac{\mathcal{P}(\mathbf{y}|\lambda_{\text{true}})}{\mathcal{P}(\mathbf{y}|\lambda_{\text{approx}})}. \tag{6.10}$$

We take $t = 6$ and find

$$\begin{aligned}
D^{(6)}_{\text{KL}}(\lambda_{\text{true}}||\lambda_{\text{SS}}) &= 0.0803, \\
D^{(6)}_{\text{KL}}(\lambda_{\text{true}}||\lambda_{\text{BW}}) &= 0.2069.
\end{aligned}$$

We conclude that the subspace inspired method performs better than the Baum-Welch approach when comparing the Kullback-Leibler divergence. As a second performance measure, we compare the likelihood of a output sequence given the true model with the likelihoods of the output sequence given the obtained models. The likelihood of a sequence $y_1 y_1 \dots y_T$ given a model λ, is defined as

$$l(\lambda) := P(y(1, 2, \dots, T) = y_1 y_1 \dots y_T | \lambda).$$

It is clear that the likelihood computed for a long sequence is typically a very small number. Therefore, we define another version of the likelihood that is better from numerical point of view

$$l^{(t)}(\lambda) := \frac{\sum_{i=1}^{T-t+1} P(y(i, i+1, \dots, i+t-1) = y_i y_{i+1} \dots y_{i+t-1} | \lambda)}{T - t + 1}. \quad (6.11)$$

We take $t = 15$ and find

$$\begin{aligned} l^{(15)}(\lambda_{\text{true}}) &= 7.6487\ 10^{-5}, \\ l^{(15)}(\lambda_{\text{SS}}) &= 6.7196\ 10^{-5}, \\ l^{(15)}(\lambda_{\text{BW}}) &= 7.3213\ 10^{-5}. \end{aligned}$$

The likelihood for Baum-Welch is closest to the likelihood computed with the true model. However, the likelihood for the subspace inspired approach is also high and close to the likelihood computed with the true model. So the subspace inspired method performs better in the Kullback-Leibler divergence between the models (Equation 6.10) and the Baum-Welch approach performs better using the likelihood performance measure (Equation 6.11). However, to our opinion, the subspace inspired method gives a good trade-off between the long-term modeling (measured by the likelihood) and the short-term modeling (measured by the Kullback-Leibler divergence) which is more important than only a good long-term modeling capacity (as was the case for Baum-Welch).

6.5 Modeling sequences from the HIV genome

In this section we use the subspace inspired identification method proposed in this chapter to model the changes observed in the DNA of a highly mutating virus: the Human Immunodeficiency Virus (HIV). Similar results were obtained in [94] using the Baum-Welch identification method. We first describe in short the mechanism behind mutations. We then describe the data and divide the data in *training data* and *test data*. Subsequently, we use the subspace inspired identification method to obtain a model of the trainings data. We check the quality of the model on test data. Finally, we explain how the model can be used to predict new viral sequences.

Mutations are changes to the nucleotide sequence of the genetic material of an organism. Gene mutations take many forms and can result in the loss of complete sections of genes, their duplications or inversions. New segments of DNA can be integrated, or genes can be broken into parts and separated. However, by far the most common type of mutations, called *point mutations*, result from the replacement of a single nucleotides within a gene. In Figure 6.3(a) we illustrate the principle of a point mutation. Radiation, exposure to certain chemicals, and some biological processes can induce mutations in genes. But even in the absence of these influences the genes of all living organisms are subject to mutations. This background mutation rate for cellular organisms is low and consequently can only be observed in organisms that reproduce in enormous numbers, like the HIV virus. In Figure 6.3(b) we schematically show the HIV virus. It had been estimated that every possible single point mutation in the HIV may occur more than 10,000 times a day in an affected person [27]. All these factors contribute to a larger number of *random* changes introduced into the viral particles that eventually lead to structural modifications. It is thus reasonable to assume that the HIV mutational processes are such that the sequences produced randomly traverse through the space of all possible sequences. In this section we therefore use hidden Markov models to model HIV sequences.

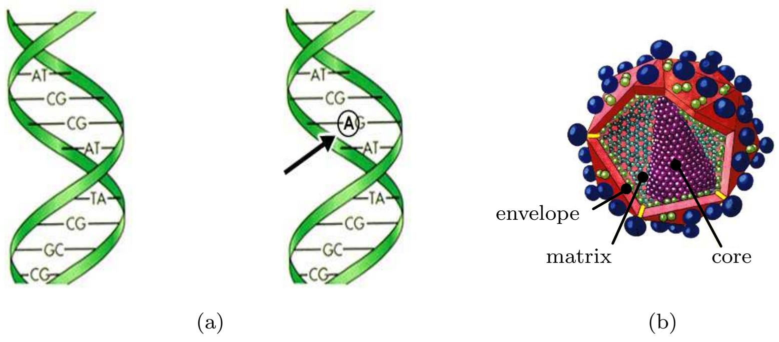


Figure 6.3: *Mutations are changes to the nucleotide sequence of the genetic material of an organism. In Subfigure (a) we illustrate the principle of a point mutation, where a single nucleotides within a gene is replaced (here the C-nucleotide is replaced by the A-nucleotide). The background mutation rate for cellular organisms is low and consequently can only be observed in organisms that reproduce in enormous numbers, like the HIV virus. In Subfigure (b) we schematically show the HIV virus. The core of the HIV virus is surrounded by a matrix composed of a viral protein. This is, in turn, surrounded by the viral envelope.*

In this section we consider 30 mutated sequence from the part of the HIV1 genome that codes for the envelope protein. The sequences have a length of 222 nucleotides. In the rest of this section we refer to these sequences with $\mathbf{u}_1, \ldots \mathbf{u}_{30}$. The data was selected from the National Center for Biotechnology Information (NCBI) database (available at `www.ncbi.nlm.nih.gov`).

We take the first 20 sequences $\mathbf{u}_1, \ldots \mathbf{u}_{20}$ as training sequences and the last 10 sequences as $\mathbf{u}_{21}, \ldots \mathbf{u}_{30}$ as test sequences. We now use Baum-Welch as well as the subspace inspired identification method proposed in this chapter to find models of the first 20 sequences of order 1, 2, 3, 4 and 5. For the subspace inspired identification method, we use $i_1 = i_2 = 4$.

To check the quality of the models, we compute the mean[1] Kullback-Leibler divergence between the Hankel block $\mathfrak{H}_{(3,3)}$ estimated from the data and the corresponding Hankel block generated by the obtained Mealy models. We show these divergences in Table 6.1. As a second performance measure, we compute the mean[2] likelihood of the given sequences (using Equation (6.11) with $t = 10$) and show these values in Table 6.2.

Table 6.1: *Mean Kullback-Leibler divergence between the Hankel block $\mathfrak{H}_{(3,3)}$ estimated from the trainings data and the corresponding Hankel block generated by the models of order 1,2,3,4 and 5 obtained by Baum-Welch identification and by the subspace inspired identification method proposed in this chapter.*

order	1	2	3	4	5
Baum-Welch	3.15	4.65	8.27	21.02	22.93
subspace	3.15	2.14	1.13	1.08	1.10

Table 6.2: *Mean likelihood of the sequences $\mathbf{u}_1, \dots \mathbf{u}_{20}$ for the models of order 1,2,3,4 and 5 obtained by Baum-Welch identification and by the subspace inspired identification proposed in this chapter.*

order	1	2	3	4	5
Baum-Welch	$8.13\ 10^{-5}$	$9.03\ 10^{-5}$	$1.40\ 10^{-4}$	$1.45\ 10^{-4}$	$1.50\ 10^{-4}$
subspace	$8.14\ 10^{-5}$	$8.84\ 10^{-5}$	$9.84\ 10^{-5}$	$9.60\ 10^{-5}$	$9.83\ 10^{-5}$

We prefer to work with the model of order 3 obtained with the subspace inspired identification procedure. It is clear from Table 6.1 and Table 6.2 that none of both performance measures improves by using an order higher than 3. Next to that, we prefer the subspace inspired approach over the Baum-Welch approach as the subspace inspired approach gives a better trade-off between long-term modeling and short-term modeling. The Baum-Welch approach only makes a long-term model.

Using this third order model, we compute the likelihoods of the 10 sequences in the test set. The likelihoods are of the same order of magnitude as the likelihoods in Table 6.2. This last observation allows us to conclude that a good model for the HIV sequences is obtained.

Now using this model a lot of *new* viral sequences can be generated. These sequences can be checked biologically.

[1]The Kullback-Leibler divergence for each of the 20 training sequences is computed and subsequently the mean of these divergences is calculated.

[2]The likelihood for each of the 20 training sequences is computed and subsequently the mean of these likelihoods is calculated.

Table 6.3: *Likelihood (calculated using Equation (6.11) with $t = 10$) of the given test sequences for the model of order 3 obtained by the subspace inspired approach proposed in this chapter. The likelihoods are of the same order of magnitude as the likelihoods in Table 6.2.*

test sequence	$\mathbf{u}_{21}$	$\mathbf{u}_{22}$	$\mathbf{u}_{23}$	$\mathbf{u}_{24}$	$\mathbf{u}_{25}$
likelihood	$9.18\ 10^{-5}$	$9.15\ 10^{-5}$	$9.26\ 10^{-5}$	$8.82\ 10^{-5}$	$9.15\ 10^{-5}$
test sequence	$\mathbf{u}_{26}$	$\mathbf{u}_{27}$	$\mathbf{u}_{28}$	$\mathbf{u}_{29}$	$\mathbf{u}_{30}$
likelihood	$8.82\ 10^{-5}$	$7.93\ 10^{-5}$	$9.01\ 10^{-5}$	$8.79\ 10^{-5}$	$9.15\ 10^{-5}$

We expect that even better models for the HIV sequences can be obtained, if prior knowledge about the structure of the sequences is incorporated into the models. In this thesis we do not go into detail about incorporating prior knowledge (see section on further research (Section 8.2)).

6.6 The linear stochastic case

In this section we compare identification for hidden Markov models with identification of linear stochastic models. There are two different types of identification methods for linear stochastic models. The first type are the optimization based methods such as prediction error methods [75], and the second type are the subspace based methods [78]. The optimization based methods are the analogue of the Baum-Welch identification for hidden Markov model, and the subspace based methods are the analogue of the method for HMMs where first the state sequence is estimated and subsequently the system matrices. In this section we review a subspace based identifcation method for linear stochastic systems.

The identification problem for linear stochastic systems can be stated as:

Problem 6.2 (linear stochastic identification problem)**.** *Given an output sequence $y_1 y_2 \dots y_T$ of length T. Find a linear stochastic model (A, C, P, Q, R, S) that models the output sequence approximately.*

Before we describe subspace based identification, we need some definitions. The *Kalman filter state sequence* of length j is defined as

$$\hat{X}_i = \left[\begin{array}{cccc} \hat{x}_i & \hat{x}_{i+1} & \dots & \hat{x}_{i+j-1} \end{array} \right],$$

where the Kalman filter estimates use only partial output information. For instance, the $(q + 1)$-th column of $\hat{X}_i$ only uses i output measurements $y_q \dots y_{i+q-1}$, instead of all output measurements up until time $i + q - 1$ (as would be expected).

Now the core idea of subspace identification, is that for a given model order n, the matrix $\hat{X}_i$ can be estimated from the output data $y_1 y_2 \dots y_T$ only. For details, we refer to [78].

In a next step the system matrices A and C can be calculated from the following least squares problem

$$\begin{bmatrix} \hat{X}_{i+1} \\ Y_{i|i} \end{bmatrix} = \begin{bmatrix} A \\ C \end{bmatrix} \hat{X}_i,$$

where $Y_{i|i} = \begin{bmatrix} y_i & y_{i+1} & \dots y_{i+j-1} \end{bmatrix}$. The system matrices Q, R and S can be calculated from the residuals of the least squares problem. It should be clear that the identification approach for hidden Markov models of Section 6.3, is analogous to the subspace approach for linear stochastic systems.

6.7 Conclusions

In this chapter the identification problem for hidden Markov models is considered. First, the Baum-Welch algorithm is derived for hidden Markov models of Mealy type. Subsequently, a new identification method for hidden Markov models is proposed. It is inspired by subspace identification for linear stochastic models. The method first estimates the state sequence directly from the output sequence and subsequently the system matrices are estimated from the obtained state sequence and the given output sequence. The subspace inspired method is applied to the modeling of sequences from the HIV genome.

Chapter 7

Recursive estimation using quasi hidden Markov models

The state estimation problem for hidden Markov models can be stated as: given output measurements up to a certain time instant τ, estimate the state at time instant t. Depending on whether $\tau > t$, $\tau = t$ or $\tau < t$, the estimation problem is a smoothing, filtering or prediction problem. Estimation problems are often used in real time applications, and therefore it is often important to have recursive solutions to the estimation problems. The recursive state filtering and prediction problems for hidden Markov models have been considered in [87]. For the smoothing problem only a non-recursive method has been considered in literature [49]. In this chapter we present a solution to the recursive state smoothing problem for hidden Markov models.

In the output smoothing problem for hidden Markov models with two output processes, measurements of the first output are given up to a certain time instant τ and the goal is to estimate the second output at time instant t. As with state estimation problems a distinction is made between the output smoothing problem, the output filtering problem and the output prediction problem. In this chapter we give solutions to the different output estimation problems. We also prove that for the output estimation problems it suffices to have a quasi hidden Markov model instead of a positive hidden Markov model. This observation gives much advantage in practical applications: first of all, it should be clear from the previous chapters that a quasi HMM can be obtained more easily than a positive HMM. Next, the order of a quasi realization is typically smaller than the order of an equivalent positive realization which makes the estimation calculations less complex.

Estimation problems for linear stochastic models have been considered in [6] and a more recent overview of the current state-of-the-art is given in [53].

List of own contributions

We here describe our contributions to the estimation problem for hidden Markov models.

- In Section 7.2.2 we show that it suffices for the output filtering and prediction problem to have a quasi hidden Markov model instead of a positive hidden Markov model.
- In Section 7.3.2 we derive formulas for the recursive fixed-point and the recursive fixed-lag state smoothing problem. In Section 7.3.1 we show that it suffices for the fixed-point and the fixed-lag output smoothing problem to have a quasi hidden Markov model instead of a positive hidden Markov model.
- In Section 7.4.2 we present a technique to solve the fixed-interval output smoothing problem.
- In Section 7.5 we apply the methods of this chapter to the coin flipping problem of Section 5.5. Given an output sequence of coin flipping experiments generated with a fair and a false coin alternatingly, we find out at which experiments the fair or the false coin was used.
- In Section 7.6 we introduce switched hidden Markov models and provide a method to determine the operation mode of a switched hidden Markov model based on the estimation methods described in this chapter.
- In Section 7.7 we apply the method to determine the operation mode in an output sequence of a switched HMM to the problem of finding motifs in DNA sequences.

Section-by-section overview

In Section 7.1 we formally introduce the different state and output estimation problems and we also define some notation. In Section 7.2 the output and state filtering and prediction problems are discussed. Subsequently, in Section 7.3 we consider the fixed-point and fixed-lag output and state smoothing problems. In Section 7.4 the fixed interval state and output smoothing problems are considered. In Section 7.5 we apply the estimation techniques to a coin flipping experiment. In Section 7.6 we consider the problem of separating the output sequence of a switched hidden Markov model and apply it to the problem of finding motifs in DNA sequences (Section 7.7). In Section 7.8 we briefly review the filtering problem for linear stochastic models and compare this with the situation for hidden Markov models.

7.1 Introduction

In this section we introduce the two types of estimation problems considered in this chapter: the state estimation problem and the output estimation problem.

More particularly, we define the filtering, prediction and smoothing problems.

Definition 7.1. *Given a Mealy HMM* $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$ *and measurements* $\mathbf{y} = y_1 y_2 \ldots$ *of the output. The* state estimation problem *consists in computing the probability distribution of the state at time instant t based on the measurements* $y_1 y_2 \ldots y_\tau$ *up to time instant* τ*. In case* $t < \tau$*, the estimation problem is called a* (state) smoothing problem. *If* $t = \tau$*, it is called a* (state) filtering problem. *If* $t > \tau$*, it is called a* (state) prediction problem. *If* $t = \tau + 1$*, we speak about a* one step ahead (state) prediction problem.

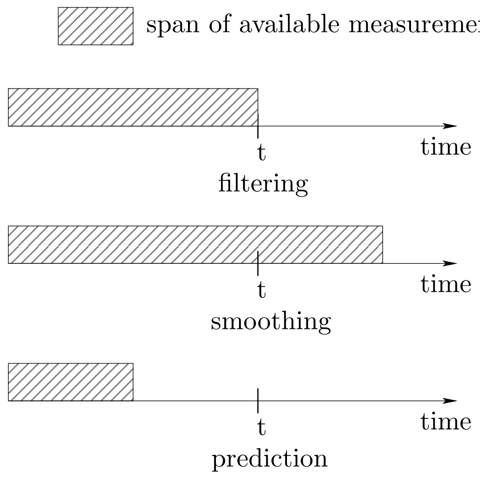


Figure 7.1: *Filtering, smoothing, prediction: the index t denotes the time at which the state/output is estimated.*

The filtering, prediction and smoothing problems are shown schematically in Figure 7.1. The filtering problem is frequently employed in real-time applications. At a certain time instant, measurements $y_1 y_2 \ldots y_t$ are available and the problem is to estimate the distribution of the state at time t. At the next time instant, the measurement y_{t+1} becomes available and the problem is to estimate the next distribution of the state at time $t+1$. An analogous situation holds for the prediction problem. The smoothing problem on the other hand can be used either real-time or offline. For the real-time smoothing problem, we distinguish between *fixed-point smoothing* and *fixed-lag smoothing.* In the fixed-point smoothing problem, at a certain time instant, one estimates the distribution of the state at time τ based on measurements up to time t, $\tau < t$. At the next time instant, one estimates again the distribution of the state at time τ but now based on measurements up to time $t + 1$. In the fixed-lag smoothing problem, one estimates the distribution of the state at time t based on measurements up to time $t + N$. At the next time instant, one estimates the distribution of the state at time $t + 1$ based on measurements up to time $t+N+1$. We also consider an offline smoothing problem called the *fixed-interval* *smoothing problem.* In this problem measurements $\mathbf{y} = y_1y_2\dots y_t$ up to time t are available, and the goal is to compute the distribution of the state sequence $x(1), x(2), \dots x(t)$. This problem is computational expensive, hence one usually concentrates on calculating the most probable state sequence $\hat{\mathbf{x}}$.

For the definition of the output estimation problem, we first need to define a Mealy HMM with two output processes. This definition was already given in Chapter 5, but we recall it here. A Mealy HMM with output processes y and z is defined as $(\mathbb{X}, \mathbb{Y}\times\mathbb{Z}, \Pi, \pi(1))$. Notice that the output alphabet of the process z is denoted by $\mathbb{Z}$ (not to be confused with the set of integers). Notice that Π is a mapping of the form $\Pi : \mathbb{Y}\times\mathbb{Z} \mapsto \mathbb{R}_+^{|\mathbb{X}|\times|\mathbb{X}|}$.

Definition 7.2. *Given a Mealy HMM with two output processes* $(\mathbb{X}, \mathbb{Y}\times\mathbb{Z}, \Pi, \pi(1))$ *and measurements* $\mathbf{y} = y_1, y_2, \dots$ *of the y-output. The* output estimation problem *consists in computing the probability distribution of the output z at time instant t based on the measurements* $y_1y_2\dots y_\tau$ *up to time instant* τ*. In case* $t < \tau$*, the estimation problem is called an* (output) smoothing problem*. If* $t = \tau$*, it is called an* (output) filtering problem*. If* $t > \tau$*, it is called an* (output) prediction problem*. If* $t = \tau + 1$*, we call this problem a* one step ahead (output) prediction problem*.*

The same remarks concerning the real-time or offline application of the state estimation problems hold for output estimation problems. In the remainder of this section, we introduce some notation that is needed in the next sections of this chapter.

First of all, $\tilde{\pi}(\tau_1; y_1, y_2, ..., y_{\tau_2})$ and $\pi(\tau_1|y_1, y_2, ..., y_{\tau_2})$ are row vectors in $\mathbb{R}^{|\mathbb{X}|}$ where

$$\begin{aligned}\tilde{\pi}_i(\tau_1; y_1, y_2, ..., y_{\tau_2}) &:= P(x(\tau_1) = i, y(1) = y_1, y(2) = y_2, ..., y(\tau_2) = y_{\tau_2}),\\ \pi_i(\tau_1|y_1, y_2, ..., y_{\tau_2}) &:= P(x(\tau_1) = i|y(1) = y_1, y(2) = y_2, ..., y(\tau_2) = y_{\tau_2}),\end{aligned}$$

and $\tilde{\omega}(\tau_1; y_1, y_2, ..., y_{\tau_2})$ and $\omega(\tau_1|y_1, y_2, ..., y_{\tau_2})$ are mappings from the output space $\mathbb{Z}$ to $\mathbb{R}_+$ where

$$\begin{aligned}\tilde{\omega}(\tau_1; y_1, y_2, ..., y_{\tau_2})(\mathtt{z}) &:= P(z(\tau_1) = \mathtt{z}, y(1) = y_1, y(2) = y_2, ..., y(\tau_2) = y_{\tau_2}),\\ \omega(\tau_1|y_1, y_2, ..., y_{\tau_2})(\mathtt{z}) &:= P(z(\tau_1) = \mathtt{z}|y(1) = y_1, y(2) = y_2, ..., y(\tau_2) = y_{\tau_2}).\end{aligned}$$

Next, $\Pi(\mathtt{y}^{(1)}, \mathtt{y}^{(2)}, ..., \mathtt{y}^{(\tau)})$ and $\Pi([\mathtt{y}^{(1)}\mathtt{z}^{(1)}]^\top, \mathtt{y}^{(2)}, ..., \mathtt{y}^{(\tau)})$ are matrices in $\mathbb{R}^{|\mathbb{X}|\times|\mathbb{X}|}$ defined as

$$\begin{aligned}&\Pi_{ij}\left(\mathtt{y}^{(1)}, \mathtt{y}^{(2)}, ..., \mathtt{y}^{(\tau)}\right) :=\\ &\quad P\left(x(t+\tau) = j, y(t) = \mathtt{y}^{(1)}, y(t+1) = \mathtt{y}^{(2)}, ..., y(t+\tau-1) = \mathtt{y}^{(\tau)}|x(t) = i\right),\\ &\Pi_{ij}\left(\begin{bmatrix}\mathtt{y}^{(1)}\\ \mathtt{z}^{(1)}\end{bmatrix}, \mathtt{y}^{(2)}, ..., \mathtt{y}^{(\tau)}\right) :=\\ &\quad P\left(x(t+\tau) = j, \begin{bmatrix}y(t)\\ z(t)\end{bmatrix} = \begin{bmatrix}\mathtt{y}^{(1)}\\ \mathtt{z}^{(1)}\end{bmatrix}, y(t+1) = \mathtt{y}^{(2)}, ..., y(t+\tau-1) = \mathtt{y}^{(\tau)}|x(t) = i\right).\end{aligned}$$

Finally, we need some derived system matrices. Given a Mealy HMM $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$, define β as a mapping from $\mathbb{Y}$ to $\mathbb{R}^{|\mathbb{X}|}$, where $\beta_i(\mathtt{y}) = P(y(t) = \mathtt{y}|x(t) = i)$. Clearly $\beta(\mathtt{y}) = \Pi(\mathtt{y})e$. Given a Mealy HMM $(\mathbb{X}, \mathbb{Y} \times \mathbb{Z}, \Pi, \pi(1))$ with output processes y and z, define

- $\beta^{(y)}$ as a mapping from $\mathbb{Y}$ to $\mathbb{R}^{|\mathbb{X}|}$, where $\beta_i^{(y)}(\mathtt{y}) = P(y(t) = \mathtt{y}|x(t) = i)$. This can be calculated as $\beta^{(y)}(\mathtt{y}) = \sum_{\mathtt{z}} \Pi(\mathtt{y}, \mathtt{z})e$. The mappings $\beta^{(z)}$ and $\beta^{(y,z)}$ are defined analogously.
- $\Pi^{(y)}$ as a mapping from $\mathbb{Y}$ to $\mathbb{R}^{|\mathbb{X}|\times|\mathbb{X}|}$, where $\Pi_{ij}^{(y)}(\mathtt{y}) := P(x(t+1) = j, y(t) = \mathtt{y}|x(t) = i)$. This can be calculated as $\Pi^{(y)}(\mathtt{y}) = \sum_{\mathtt{z}} \Pi(\mathtt{y}, \mathtt{z})$.

7.2 Filtering and prediction

In this section we give a solution to the recursive filtering and prediction problem. We make a distinction between state filtering/prediction (Section 7.2.1) and output filtering/prediction (Section 7.2.2). It is shown that for the output estimation case, it suffices to have a quasi HMM instead of a positive HMM.

7.2.1 State filtering and state prediction

The algorithm below gives a solution to the recursive state filtering and prediction problem for HMMs.

Algorithm 7.1. *Given a Mealy HMM* $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$. *The following equations define a recursive algorithm to compute* $\pi(t|y_1, y_2, ..., y_{t-1})$ *and* $\pi(t|y_1, y_2, ..., y_t)$*:*

$$\begin{aligned}
\tilde{\pi}(1) &= \pi(1), \\
\tilde{\pi}(t+1; y_1, y_2, ..., y_t) &= \tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\Pi(y_t), \\
\pi(t|y_1, y_2, ..., y_{t-1}) &= \frac{\tilde{\pi}(t; y_1, y_2, ..., y_{t-1})}{\tilde{\pi}(t; y_1, y_2, ..., y_{t-1})e} \\
\pi(t|y_1, y_2, ..., y_t) &= \frac{\tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\,\mathrm{diag}(\beta^{(y)}(y_t))}{\tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\beta^{(y)}(y_t)}
\end{aligned}$$

The proof of the algorithm follows immediately from calculation with probabilities. The details are omitted. For a Moore HMM $(\mathbb{X}, \mathbb{Y}, \Pi_{\mathbb{X}}, \beta, \pi(1))$, the computation of $\tilde{\pi}(t+1; y_1, y_2, ..., y_t)$ from $\tilde{\pi}(t; y_1, y_2, ..., y_{t-1})$ can be split up in two steps: a *measurement update step* (7.1) and a *time update step* (7.2).

$$\begin{aligned}
\tilde{\pi}(t; y_1, y_2, ..., y_t) &= \tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\,\mathrm{diag}(\beta(y_t)), && (7.1) \\
\tilde{\pi}(t+1; y_1, y_2, ..., y_t) &= \tilde{\pi}(t; y_1, y_2, ..., y_t)\Pi_{\mathbb{X}}. && (7.2)
\end{aligned}$$

The computation of $\tilde{\pi}(t+1; y_1, y_2, ..., y_t)$ from $\tilde{\pi}(t; y_1, y_2, ..., y_{t-1})$ for a Mealy HMM can also be split up into two steps. However, in both steps the measurement y_t is needed, hence the splitting does not yield any advantage.

7.2.2 Output filtering and output prediction

In Algorithm 7.2 we describe a recursive algorithm to solve both the output filtering and the output prediction problem. In Proposition 7.1 we show that a similar algorithm can be used when a quasi hidden Markov model is given instead of a positive hidden Markov model.

Algorithm 7.2. *Given is a hidden Markov model* $(\mathbb{X}, \mathbb{Y} \times \mathbb{Z}, \Pi, \pi(1))$*. The following equations define a recursive algorithm that computes* $\omega(t|y_1, y_2, ..., y_{t-1})$ *and* $\omega(t|y_1, y_2, ..., y_t)$*:*

$$\begin{aligned}
\tilde{\pi}(1) &= \pi(1),\\
\tilde{\pi}(t+1; y_1, y_2, ..., y_t) &= \tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\Pi^{(y)}(y_t),\\
\omega(t|y_1, y_2, ..., y_{t-1})(\boldsymbol{z}) &= \frac{\tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\beta^{(z)}(\boldsymbol{z})}{\tilde{\pi}(t; y_1, y_2, ..., y_{t-1})e}\\
\omega(t|y_1, y_2, ..., y_t)(\boldsymbol{z}) &= \frac{\tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\beta^{(y,z)}(y_t, \boldsymbol{z})}{\tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\beta^{(y)}(y_t)}
\end{aligned}$$

The proof of the algorithm follows from calculation with probabilities and is omitted. However, we prove an interesting property of the algorithm.

Proposition 7.1. *Given a quasi hidden Markov model* $(\mathbb{Q}, \mathbb{Y} \times \mathbb{Z}, A, c, b)$*. The recursive output filtering and prediction problem is solved using Algorithm 7.2, where the system matrices are replaced by quasi system matrices, i.e.* $(\Pi, \pi(1), e) \to (A, c, b)$*, and the derived system matrices are replaced by quasi derived system matrices, i.e.* $(\beta^{(y)}, \beta^{(z)}, \beta^{(y,z)}, \Pi^{(y)}) \to (\lambda^{(y)}, \lambda^{(z)}, \lambda^{(y,z)}, A^{(y)})$*. Quasi derived system matrices are computed from the quasi system matrices in the same way as the derived system matrices are computed from the system matrices.*

Proof: For the filtering case, this can be seen from

$$\begin{aligned}
\omega(t|y_1, y_2, ..., y_t)(\mathsf{z}) &= \frac{P(y(1) = \mathrm{y}_1, y(2) = \mathrm{y}_2, ..., y(t) = \mathrm{y}_t, z(t) = \mathsf{z})}{P(y(1) = \mathrm{y}_1, y(2) = \mathrm{y}_2, ..., y(t) = \mathrm{y}_t)}\\
&= \frac{\tilde{c}(t; y_1, y_2, ..., y_{t-1})\lambda^{(z)}(\mathsf{z})}{\tilde{c}(t; y_1, y_2, ..., y_{t-1})b},
\end{aligned}$$

where $\tilde{c}(t; y_1, y_2, ..., y_{t-1})$ is the quasi equivalent of $\tilde{\pi}(t; y_1, y_2, ..., y_{t-1})$. The proof for the prediction case is analogous. ∎

When the filter is used with a positive realization, the intermediate variable $\tilde{\pi}(t+1|y_1, y_2, ..., y_t)$ has an interpretation. It is proportional to the probability distribution of the state at time instant $t+1$ given output measurements up to time instant t (see Algorithm 7.1). When using a quasi realization, the intermediate variable $\tilde{c}(t+1|y_1, y_2, ..., y_t)$ does not have an interpretation anylonger. However, working with a quasi realization has important advantages. A quasi realization is easy to compute and in addition a quasi realization typically has lower order than a positive realization which makes the estimator less complex.

7.3 Fixed-point and fixed-lag smoothing

In this section we consider the recursive fixed-point and fixed-lag smoothing problem. We make a distinction between state smoothing (Section 7.3.1) and output smoothing (Section 7.3.2). Again, it can be shown that for the output smoothing case, it suffices to have a quasi HMM instead of a positive HMM.

7.3.1 Fixed-point and fixed-lag state smoothing

The algorithm below provides a solution to the recursive filtering and prediction problem for HMMs.

Algorithm 7.3. *Given a hidden Markov model* $(\mathbb{X}, \mathbb{Y}, \Pi, \pi(1))$. *Then the following equations define a recursive algorithm that computes* $\pi(t|y_1, y_2, ..., y_\tau)$ *and* $\pi(t|y_1, y_2, ..., y_{t+N})$*:*

$$\begin{aligned}
\tilde{\pi}(1) &= \pi(1), \\
\tilde{\pi}(t+1; y_1, y_2, ..., y_t) &= \tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\Pi(y_t), \\
\Pi(y_t) &= \Pi(y_t), \\
\Pi(y_t, y_{t+1}, ..., y_k) &= \Pi(y_t, y_{t+1}, ..., y_{k-1})\Pi(y_k), \\
\tilde{\pi}(t; y_1, y_2, ..., y_\tau) &= \tilde{\pi}(t; y_1, y_2, ..., y_{t-1}) \operatorname{diag}(\Pi(y_t, y_{t+1}, ..., y_\tau)e), \\
\pi(t|y_1, y_2, ..., y_\tau) &= \frac{\tilde{\pi}(t; y_1, y_2, ..., y_\tau)}{\tilde{\pi}(t; y_1, y_2, ..., y_\tau)e}, \\
\tilde{\pi}(t; y_1, y_2, ..., y_{t+N}) &= \tilde{\pi}(t; y_1, y_2, ..., y_{t-1}) \operatorname{diag}(\Pi(y_t, y_{t+1}, ..., y_{t+N})e), \\
\pi(t|y_1, y_2, ..., y_{t+N}) &= \frac{\tilde{\pi}(t; y_1, y_2, ..., y_{t+N})}{\tilde{\pi}(t; y_1, y_2, ..., y_{t+N})e}.
\end{aligned}$$

The proof of the algorithm follows from calculation with probabilities and is omitted.

7.3.2 Fixed-point and fixed-lag output smoothing

In Algorithm 7.4 we describe a recursive algorithm to solve both the recursive fixed-point and the fixed-lag output smoothing problem. In Proposition 7.2 we show that a similar algorithm can be used when a quasi hidden Markov model is given instead of a positive hidden Markov model.

Algorithm 7.4. *Given a hidden Markov model* $(\mathbb{X}, \mathbb{Y} \times \mathbb{Z}, \Pi, \pi(1))$. *Then the following equations define a recursive algorithm that computes* $\omega(t|y_1, y_2, ..., y_\tau)$ *and* $\omega(t|y_1, y_2, ..., y_{t+N})$*:*

$$\begin{aligned}
\tilde{\pi}(1) &= \pi(1), \\
\tilde{\pi}(t+1; y_1, y_2, ..., y_t) &= \tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\Pi^{(y)}(y_t), \\
\Pi([y_t, \boldsymbol{z}]^\top) &= \Pi(y_t, \boldsymbol{z}), \\
\Pi([y_t, \boldsymbol{z}]^\top, y_{t+1}, ..., y_k) &= \Pi([y_t, \boldsymbol{z}]^\top, y_{t+1}, ..., y_{k-1})\Pi^{(y)}(y_k),
\end{aligned}$$

$$\begin{aligned}
\tilde{\omega}(t; y_1, y_2, ..., y_\tau)(\boldsymbol{z}) &= \tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\Pi([y_t, \boldsymbol{z}]^\top, y_{t+1}, ..., y_\tau)e,\\
\omega(t|y_1, y_2, ..., y_\tau)(\boldsymbol{z}) &= \frac{\tilde{\omega}(t; y_1, y_2, ..., y_\tau)(\boldsymbol{z})}{\sum_{\boldsymbol{z}} \tilde{\omega}(t; y_1, y_2, ..., y_\tau)(\boldsymbol{z})},\\
\tilde{\omega}(t; y_1, y_2, ..., y_{t+N})(\boldsymbol{z}) &= \tilde{\pi}(t; y_1, y_2, ..., y_{t-1})\Pi([y_t, \boldsymbol{z}]^\top, y_{t+1}, ..., y_{t+N})e,\\
\omega(t; y_1, y_2, ..., y_{t+N})(\boldsymbol{z}) &= \frac{\tilde{\omega}(t; y_1, y_2, ..., y_{t+N})(\boldsymbol{z})}{\sum_{\boldsymbol{z}} \tilde{\omega}(t; y_1, y_2, ..., y_{t+N})(\boldsymbol{z})}.
\end{aligned}$$

We omit the proof the Algorithm 7.4, but prove an interesting property of the algorithm.

Proposition 7.2. *Given a quasi hidden Markov model $(\mathbb{Q}, \mathbb{Y} \times \mathbb{Z}, A, c, b)$. The recursive fixed-point and fixed-lag smoothing problem is solved using Algorithm 7.4, where the system matrices are replaced by quasi system matrices and the derived system matrices are replaced by quasi derived system matrices.*

Proof: The proof is analogous to the proof of Proposition 7.1. ■

7.4 Fixed-interval smoothing

In this section we consider the fixed-interval smoothing problem. We make a distinction between state smoothing (Section 7.4.1) and output smoothing (Section 7.4.2).

7.4.1 Fixed-interval state smoothing

In the fixed interval smoothing problem, the goal is to estimate the distribution of the state $x(1), x(2), \ldots, x(t)$ from measurements $y_1 y_2 \ldots y_t$ of y up to time t. Define $\pi(1, 2, \ldots, t|y_1, y_2, \ldots, y_t)$ as a tensor in $[0, 1]^{|\mathbb{X}|^t}$ with

$$\pi(1, 2, \ldots, t|y_1, y_2, \ldots, y_t)_{i_1, i_2, \ldots, i_t} = \frac{P\begin{pmatrix} y(1) = y_1, \ldots, y(t) = y_t \\ x(1) = i_1, \ldots, x(t) = i_t \end{pmatrix}}{\mathcal{P}(y_1 y_2 \ldots y_t)}. \quad (7.3)$$

Calculating the complete distribution of $x(1), x(2), \ldots, x(t)$ is computational expensive. However, in many applications, one is not interested in knowing the complete distribution of the state sequence x, buth rather in the most probable state sequence $\hat{\mathbf{x}}$. The Viterbi algorithm [49, 115] computes the most probable sequence $\hat{\mathbf{x}}$ without computing (7.3) for every possible $i_1 i_2 \ldots i_t \in \mathbb{X}^t$. We briefly review the Viterbi algorithm. The goal is to find $\hat{\mathbf{x}}$ as

$$\hat{\mathbf{x}} = \underset{i_1 i_2 \ldots i_t \,\in\, \mathbb{X}^t}{\operatorname{argmax}} \pi(1, 2, ..., t|y_1, y_2, ..., y_t)_{i_1, i_2, \ldots, i_t},$$

or equivalently

$$\hat{\mathbf{x}} = \underset{i_1 i_2 \ldots i_t \,\in\, \mathbb{X}^t}{\operatorname{argmax}} P\begin{pmatrix} y(1) = y_1, \ldots, y(t) = y_t \\ x(1) = i_1, \ldots, x(t) = i_t \end{pmatrix},$$

$$= \underset{i_1 i_2 \dots i_t \;\in\; \mathbb{X}^t}{\text{argmax}} \; \pi_{i_1}(1)\Pi_{i_1,i_2}(y_1)\Pi_{i_2,i_3}(y_2)\dots\Pi_{i_t,:}(y_t)e.$$

Now define

$$\begin{aligned} U(i_1, i_2, \dots, i_t) &= -\log\left(\pi_{i_1}(1)\Pi_{i_1,i_2}(y_1)\Pi_{i_2,i_3}(y_2)\dots\Pi_{i_t,:}(y_t)e\right), \\ &= -\left(\log(\pi_{i_1}(1)) + \sum_{k=1}^{t-1}\log(\Pi_{i_k,i_{k+1}}(y_k)) + \log(\Pi_{i_t,:}(y_t)e)\right), \end{aligned}$$

then

$$\hat{\mathbf{x}} = \underset{i_1 i_2 \dots i_t \;\in\; \mathbb{X}^t}{\text{argmin}} \; U(i_1, i_2, \dots, i_t).$$

This reformulation enables us to view terms like $\log(\Pi_{i_k,i_{k+1}}(y_k))$ as the cost associated in going from state i_k to i_{k+1} at time k. On the other hand $\log(\pi_{i_1}(1))$ is the cost of starting in the state i_1 at time instant 1, while $\log(\Pi_{i_t,:}(y_t)e)$ is the cost of ending in the state i_t at time instant t. In addition, the cost for going over a sequence of states is equal to the sum of the individual costs for going from one state to another. Now finding the optimal state path is merely a matter of finding a path (i.e. sequence of states $\hat{\mathbf{x}}$) of minimum cost through which the observation sequence $y_1 y_2 \dots y_t$ occurs. The Viterbi algorithm is a dynamic programming approach for finding the path of minimal cost. Below we give the Viterbi algorithm for Mealy HMMs. In literature the Viterbi algorithm is usually presented for Moore models.

Algorithm 7.5. *Given measurements $y_1 y_2 \dots y_t$ of the process y up to time instant t. Perform the following steps.*

1. *Initialization*

$$\begin{aligned} \delta_1(i) &= -\log(\pi_i(1)), \quad i \in \mathbb{X}, \\ \psi_1(i) &= 0. \end{aligned}$$

2. *Recursive computation*

$$\begin{aligned} \delta_k(j) &= \min_{i\in\mathbb{X}} \delta_{k-1}(i) - \log(\Pi_{ij}(y_{k-1})), \qquad 1 < k \le t;\; j \in \mathbb{X}, \\ \psi_k(j) &= \underset{i\in\mathbb{X}}{\text{argmin}}\, \delta_{k-1}(i) - \log(\Pi_{ij}(y_{k-1})), \qquad 1 < k \le t;\; j \in \mathbb{X}. \end{aligned}$$

3. *Termination*

$$\hat{x}_t = \underset{i\in\mathbb{X}}{\text{argmin}}\, \delta_t(i) - \log(\Pi_{i,:}(y_t)e).$$

4. *Tracing back the optimal state sequence ($k = t-1, t-2, \dots, 1$)*

$$\hat{x}_k = \psi_{k+1}(\hat{x}_{k+1}).$$

7.4.2 Fixed-interval output smoothing

In the fixed interval smoothing problem for estimating output z from output y, the goal is to estimate the distribution of the output $z(1), z(2), ..., z(t)$ from measurements $y_1 y_2 ... y_t$ of y up to time instant t. One can compute $\omega(1, 2, ..., t | y_1, y_2, ..., y_t)(\mathtt{z}^{(1)}, \mathtt{z}^{(2)}, ..., \mathtt{z}^{(t)})$ as

$$\omega(1, 2, ..., t | y_1, y_2, ..., y_t)(\mathtt{z}^{(1)}, \mathtt{z}^{(2)}, ..., \mathtt{z}^{(t)}) \quad = \quad \frac{\mathcal{P}\begin{pmatrix} y_1 & y_2 & \cdots & y_t \\ \mathtt{z}^{(1)} & \mathtt{z}^{(2)} & \cdots & \mathtt{z}^{(t)} \end{pmatrix}}{\mathcal{P}(y_1 y_2 ... y_t)}. \qquad (7.4)$$

Computing the complete distribution (7.4) is computationaly expensive. However, in many applications one is not interested in knowing the complete distribution of $z(1), z(2), ..., z(t)$, but only in the most probable output sequence $\hat{\mathbf{z}} = \hat{z}_1 \hat{z}_2 \ldots \hat{z}_t$.

One might think that the most probable output sequence $\hat{\mathbf{z}}$ from measurements of the output sequence up to time t can be computed using the following two step procedure. In the first step, one uses the Viterbi algorithm to compute the most probable state sequence from the observed output sequence, and in the second step, one determines $\hat{z}_k$ as the most probable symbol from the set $\mathbb{Z}$ corresponding to the state $\hat{x}_k$, i.e. $\hat{z}_k = \operatorname{argmax}_{\mathtt{z}} \beta^{(z)}_{\hat{x}_k}(\mathtt{z})$. However one can easily see that this approach does not give rise to the most probable sequence $\hat{\mathbf{z}}$ as desired.

We now explain how the sequence $\hat{\mathbf{z}}$ can be computed without computing (7.4) for every possible $\mathtt{z}^{(1)}\mathtt{z}^{(2)}...\mathtt{z}^{(t)} \in \mathbb{Z}^t$. The goal is to solve

$$\hat{\mathbf{z}} \quad = \quad \underset{\mathtt{z}^{(1)}\mathtt{z}^{(2)}...\mathtt{z}^{(t)} \,\in\, \mathbb{Z}^t}{\operatorname{argmax}} \quad \omega(1, 2, ..., t | y_1, y_2, ..., y_t)(\mathtt{z}^{(1)}, \mathtt{z}^{(2)}, ..., \mathtt{z}^{(t)}), \qquad (7.5)$$

or equivalently

$$\hat{\mathbf{z}} \quad = \quad \underset{\mathtt{z}^{(1)}\mathtt{z}^{(2)}...\mathtt{z}^{(t)} \,\in\, \mathbb{Z}^t}{\operatorname{argmax}} \quad \mathcal{P}\begin{pmatrix} y_1 & y_2 & \cdots & y_t \\ \mathtt{z}^{(1)} & \mathtt{z}^{(2)} & \cdots & \mathtt{z}^{(t)} \end{pmatrix}.$$

The method computes $\mathcal{P}\begin{pmatrix} y_1 & y_2 & \cdots & y_t \\ \mathtt{z}^{(1)} & \mathtt{z}^{(2)} & \cdots & \mathtt{z}^{(t)} \end{pmatrix}$ for $\mathtt{z}^{(1)}\mathtt{z}^{(2)} \ldots \mathtt{z}^{(t)} \in \mathbb{Z}^t$ in a structured way which makes it possible to discover in advance that certain sequences can impossibly be the most probable output sequence $\hat{\mathbf{z}}$. These sequences do not have to be considered anymore. The technique builds a tree as in Figure 7.2. The tree has $t+1$ levels: the root level (level 0); $t-1$ internal levels (level 1 to level $t-1$) and the leaf level (level t). Every node of the three has $|\mathbb{Z}|$ children. In every node of the three, a label of $\mathbb{R}_+^{1\times|\mathbb{X}|}$ is stored. For the root node, the label is equal to $\pi(1)$ and the label for the other nodes are computed from the label of their parent node. The label of the j-th child of a node at level i is equal to the label of the parent node multiplied by $\Pi(y_{i+1}, \mathtt{z}_j)$, where $\mathtt{z}_j$ is the j-th symbol from $\mathbb{Z}$ with respect to a certain ordering of the symbols of $\mathbb{Z}$. It is clear that for every leaf node, there exists a unique path

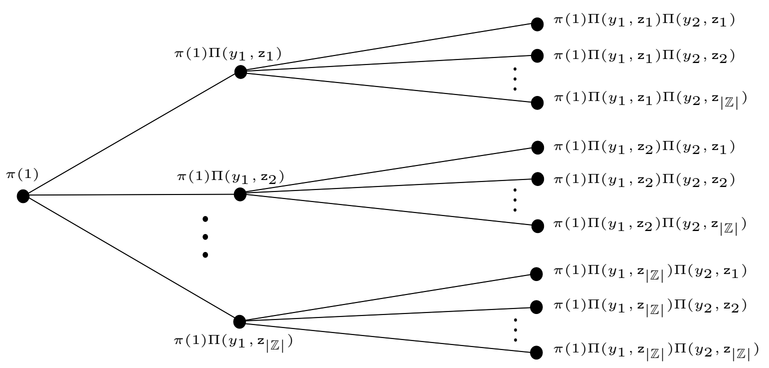

$\pi(1)$
$\pi(1)\Pi(y_1, z_1)$
$\pi(1)\Pi(y_1, z_2)$
$\pi(1)\Pi(y_1, z_{|\mathbb{Z}|})$
$\pi(1)\Pi(y_1, z_1)\Pi(y_2, z_1)$
$\pi(1)\Pi(y_1, z_1)\Pi(y_2, z_2)$
$\pi(1)\Pi(y_1, z_1)\Pi(y_2, z_{|\mathbb{Z}|})$
$\pi(1)\Pi(y_1, z_2)\Pi(y_2, z_1)$
$\pi(1)\Pi(y_1, z_2)\Pi(y_2, z_2)$
$\pi(1)\Pi(y_1, z_2)\Pi(y_2, z_{|\mathbb{Z}|})$
$\pi(1)\Pi(y_1, z_{|\mathbb{Z}|})\Pi(y_2, z_1)$
$\pi(1)\Pi(y_1, z_{|\mathbb{Z}|})\Pi(y_2, z_2)$
$\pi(1)\Pi(y_1, z_{|\mathbb{Z}|})\Pi(y_2, z_{|\mathbb{Z}|})$

Figure 7.2: *Part of the tree that needs to be built for Algorithm 7.6 to compute the most probable sequence $\hat{\mathbf{z}}$ for given measurements $y_1 y_2 \ldots y_t$.*

starting in the root and ending in that leaf node. In addition, every path from the root to a leaf node symbolizes a $\mathsf{z}^{(1)}\mathsf{z}^{(2)}...\mathsf{z}^{(t)}$-path, such that every leaf node symbolizes a possible $\mathsf{z}^{(1)}\mathsf{z}^{(2)}...\mathsf{z}^{(t)}$-path. Now by construction, the label of a leaf node multiplied with e is equal to the probability $\mathcal{P}\begin{pmatrix} y_1 & y_2 & \cdots & y_t \\ \mathsf{z}^{(1)} & \mathsf{z}^{(2)} & \ldots & \mathsf{z}^{(t)} \end{pmatrix}$ of the $\mathsf{z}^{(1)}\mathsf{z}^{(2)}...\mathsf{z}^{(t)}$-path symbolized by that leaf node.

To find the most probable path through the tree, it is not necessary to compute the probabilities at all nodes of the tree. If at a certain level of the tree, there is a node n_1 for which the label is elementwise smaller than or equal to the label of another node n_2 of the same level, then that node (and all its children, grandchildren,...) does not have to be considered anymore. Indeed, node n_1 can never be part of the most probable path as there will always be a path through node n_2 with a higher or equal probability. Below, we give an algorithm that computes $\hat{\mathbf{z}}$.

Algorithm 7.6. *Given measurements of y up to time instant t, put $i = 0$ and set $\mathcal{T}$ equal to a tree with only a root node. The root node has a label equal to $\pi(1)$ and an indicator equal to 1. Now, perform the following steps.*

1. *Add $|\mathbb{Z}|$ children to every node of level i for which the indicator is equal to 1.*

2. *Calculate the label for every node at level $i+1$ as the label of its parent multiplied by $\Pi(y_{i+1}, \boldsymbol{z}_j)$, with $\boldsymbol{z}_j$ the j-th symbol from $\mathbb{Z}$ where the considered node of level $i+1$ is the j-th child of the parent at level i.*

3. *Put the indicator of a node at level $i+1$ equal to 0 if there is a node at the same level with a label which is elementwise higher than or equal to the label of the considered node, and equal to 1 otherwise.*

4. *Increase i by 1. If $i = t$ then goto step 5. If $i < t$ goto step 1.*

5. *Find the leaf node n_* for which the label multiplied by e is highest.*
6. *The most probable sequence $\hat{\mathbf{z}}$ is the $\boldsymbol{z}^{(1)}\boldsymbol{z}^{(2)}...\boldsymbol{z}^{(t)}$-path symbolized by the leaf node n_*.*

In case a quasi realization is given, the pruning method described above does not work. One possible method to find the most probable path $\hat{\mathbf{z}}$ is by computing the whole tree. An interesting open problem is to investigate whether it is possible to compute the most probable path in an efficient way in case a quasi HMM is given.

7.5 Filtering for the coin flipping experiment

In this section we continue the example of Section 5.5. In Section 5.5 coin flipping experiments with alternatingly a fair and a false coin are considered. Given the output sequence of the experiments as well as the knowledge which coin was used at which experiment, we have built a model for the coin flipping process. In this section we consider the problem where the model is given as well as the output sequence of a series of flipping experiments. The goal is to find out which coin was used in which experiment.

This problem is a typical output prediction problem. The first output sequence is given (the outcome of the flipping experiments) and the goal is to find the second output sequence indicating which coin was used. To solve this problem, we use a modified version of the output prediction method presented in Section 7.2.2. In the method of Section 7.2.2 the prediction of the second output at time instant t is carried out based on measurements of the first output up to time instant $t-1$. The predictor used to solve the problem at hand estimates the z-output at time instant t based on measurements of y at time instants $t-N, t-N+1, \ldots, t-1$ with $N=5$. The formulas for this predictor can easily be obtained using the same approach as in Section 7.2.2.

In Figure 7.3, we show the probability (with solid line), computed with the prediction techniques, that the coin used at experiment t is the false one. Knowing the rule that Person A uses to switch between the coins, the true sequence that indicates which coin is used, can be calculated. To check the quality of the filtering algorithm, we also plot this sequence (with $*$). One can easily see that the probability that the false coin was used is high at moments where indeed the fase coin was used. We conclude that the combination of modeling and prediction works well.

7.6 Determining the operation mode of a switched HMM

In this section we show that the methods of this chapter can be used to determine the operation mode of a switched HMM. In Section 7.7 we apply these techniques to the bioinformatics problem of finding motifs in DNA sequences.

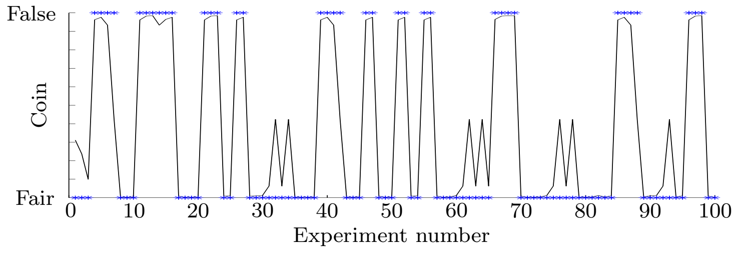


Figure 7.3: *We show the probability (with solid line), computed with the prediction techniques, that the coin used at experiment t is the false one. Knowing the rule that Person A uses to switch between the coins, the true sequence that indicates which coin is used, can be calculated. To check the quality of the filtering algorithm, we also plot this sequence (with $*$). One can easily see that the probability that the false coin was used is high at moments where indeed the fase coin was used. We conclude that the combination of modeling and prediction works well.*

We first define switched HMMs and subsequently we explain how an output sequence of a switched HMM can be separated using the filtering techniques explained in the previous sections of this chapter.

A *switched HMM* is a HMM that consists of two individual HMMs. At certain time instants, the switched HMM switches from the one HMM to the other (see Figure 7.4). Analogously, a *switched quasi HMM* is a quasi HMM that consists of two quasi HMMs. At certain time instants, it switches from the one quasi HMM to the other. We explain the notations for switched quasi HMMs. As a switched HMM is also a switched quasi HMM, the results are also valid for switched HMMs. We suppose that a switched HMM has, next to the output y, an additional output z taking values in $\mathbb{Z} = \{1, 2\}$ indicating which of the individual HMMs is active.

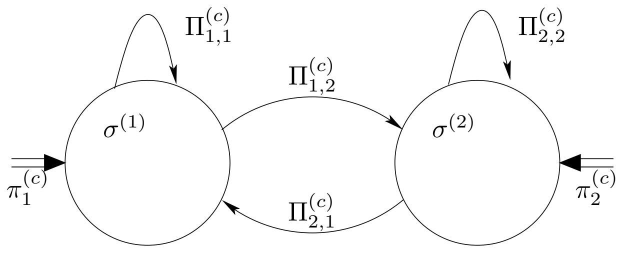


Figure 7.4: *Switched hidden Markov model.*

Consider two individual quasi HMMs $\sigma^{(1)} = (\mathbb{Y}, \mathbb{X}^{(1)}, A^{(1)}, c^{(1)}, b^{(1)})$ and $\sigma^{(2)} = (\mathbb{Y}, \mathbb{X}^{(2)}, A^{(2)}, c^{(2)}, b^{(2)})$ and assume, without loss of generality[1], that

[1]A quasi Mealy HMM $(\mathbb{Y}, \mathbb{X}, A, c, b)$ can be transformed into the equivalent HMM

$b^{(1)} = e$ and $b^{(2)} = e$. Then the switched HMM determined by the individual HMMs $\sigma^{(1)}$ and $\sigma^{(2)}$ is given by $(\mathbb{Y} \times \mathbb{Z}, \mathbb{X}^{(1)} \cup \mathbb{X}^{(2)}, A, c, e)$ where

$$\begin{aligned} A(\mathtt{y},1) &= \begin{bmatrix} \Pi^{(c)}_{1,1}A^{(1)}(\mathtt{y}) & \Pi^{(c)}_{1,2}A^{(1)}(\mathtt{y})ec^{(2)} \\ 0 & 0 \end{bmatrix}, \quad \forall \mathtt{y} \in \mathbb{Y}, \\ A(\mathtt{y},2) &= \begin{bmatrix} 0 & 0 \\ \Pi^{(c)}_{2,1}A^{(2)}(\mathtt{y})ec^{(1)} & \Pi^{(c)}_{2,2}A^{(2)}(\mathtt{y}) \end{bmatrix}, \quad \forall \mathtt{y} \in \mathbb{Y}, \\ c &= \begin{bmatrix} \pi^{(c)}_1 c^{(1)} & \pi^{(c)}_2 c^{(2)} \end{bmatrix}, \end{aligned} \tag{7.6}$$

where $\pi^{(c)}_i, i = 1,2$ is the probability that the initial model is $\sigma^{(i)}$ and $\Pi^{(c)}_{ij}, i = 1,2, j = 1,2$ is the probability to switch from $\sigma^{(i)}$ to $\sigma^{(j)}$.

Proposition 7.3. *The switched quasi HMM* $(\mathbb{Y} \times \mathbb{Z}, \mathbb{X}^{(1)} \cup \mathbb{X}^{(2)}, A, c, e)$ *defined by (7.6) is consistent.*

Proof: The proposition follows from

$$\begin{aligned} ce &= \pi^{(c)}_1 c^{(1)}e + \pi^{(c)}_2 c^{(2)}e = 1, \\ \sum_{\mathtt{y}\in\mathbb{Y},\mathtt{z}\in\mathbb{Z}} A(\mathtt{y},\mathtt{z})e &= \begin{bmatrix} \Pi^{(c)}_{1,1}A^{(1)}_{\mathbb{Q}} & \Pi^{(c)}_{1,2}ec^{(2)} \\ \Pi^{(c)}_{2,1}ec^{(1)} & \Pi^{(c)}_{2,2}A^{(2)}_{\mathbb{Q}} \end{bmatrix} e \\ &= \begin{bmatrix} \Pi^{(c)}_{1,1}e + \Pi^{(c)}_{1,2}e \\ \Pi^{(c)}_{2,1}e + \Pi^{(c)}_{2,2}e \end{bmatrix} e = e. \end{aligned}$$

■

Now in case a switched quasi HMM determined by the individual HMMs $\sigma^{(1)}$ and $\sigma^{(2)}$ is given, as well as measurements of the output y, then the output z can be determined using the techniques of Section 7.2.2 and Section 7.3.2. In case a switched positive HMM is given, the technique of Section 7.4.2 can also be used. Hence the techniques of this chapter allow to determine the operation mode of a switched HMM.

7.7 Motif detection in DNA sequences

In this section we apply the methods to determine the operation mode of a switched hidden Markov model to a problem from bioinformatics. More precisely the problem of finding motifs in DNA sequences is considered.

As already explained in Section 4.4, desoxyribonucleic acid (DNA) is a nucleic acid that contains the genetic instructions used in the development and functioning of all living organisms. DNA forms a double helix of two anti-parallel chains with complementary nucleotide sequences. The building blocks of the nucleotide sequences are the following four nucleotides: adenine (A), cytosine (C), guanine (G) and thymine (T). Certain parts of the DNA (the

$(\mathbb{Y}, \mathbb{X}, TAT^{-1}, cT^{-1}, Tb)$. Moreover, T can be chosen such that $Tb = e$.

genes) regulate the formation of certain proteins. One step in the process from DNA to protein is the binding of a certain transcription factor with the DNA. It has been shown [13], that there must exist a certain complementarity between the transcription factor and a part of the DNA for a binding to take place. A model for a part of the DNA where possibly a binding with a transcription factor can take place is called a *motif.* The parts of the DNA in between the motifs are called the *background.* An important topic in bio-informatics is the search for motifs in DNA sequences. In this section we consider this problem where the motif is specified by a position weight matrix.

We first define a position weight matrix of a motif. Subsequently, we explain how to build a hidden Markov model of the motif and the background separately. Next, we describe how the switched hidden Markov model of the motif and background together is built. Subsequently the methods are applied to the specific example of detecting motifs in muscle-specific genes.

A *Position Weight Matrix (PWM)* of a motif of length w is given by

$$P = \begin{bmatrix} P_{A,1} & P_{A,2} & \dots & P_{A,w} \\ P_{C,1} & P_{C,2} & \dots & P_{C,w} \\ P_{G,1} & P_{G,2} & \dots & P_{G,w} \\ P_{T,1} & P_{T,2} & \dots & P_{T,w} \end{bmatrix} \tag{7.7}$$

where $P_{X,k}$ is the probability to find the nucleotide X at position k of the motif. Notice that this model is a *time-inhomogeneous static model.* The model is static because the event of observing a certain nucleotide at position i is independent of the nucleotide at position j for $i \neq j$, $i \leq w$ and $j \leq w$. The model is time-inhomogeneous because the probability of observing a certain symbol differs from position to position. One could think of using a dynamic (either time-homogeneous or time-inhomogeneous) model instead of the PWM model. Although useful, this would need more training data than is usually available and it has been shown to give only slightly better results [77]. For that reason, one typically works with PWM models for modeling motifs.

A HMM representation of a PWM model is of the form presented in Figure 7.5. Note that such HMM produces a string of length $w < \infty$. This in contrast with the HMMs defined before that produce an output string of infinite length. However, we show that by combining this *finite length HMM* with the quasi HMM of the background, we obtain a switched quasi HMM of the classical sense, i.e. producing an infinite output sequence. The finite length Moore HMM of the PWM of Equation (7.7), is given by $(\mathbb{X}^{(\mathrm{m})}, \mathbb{Y}, \Pi_{\mathbb{X}}^{(\mathrm{m})}, \beta^{(\mathrm{m})}, \pi^{(\mathrm{m})}(1))$, where $\mathbb{X}^{(\mathrm{m})} = \{1, 2, \dots, w\}$, $\mathbb{Y} = \{A, C, G, T\}$, $\pi^{(\mathrm{m})}(1) = [1 \ 0 \ \dots \ 0]$ and

$$\begin{aligned} \Pi_{\mathbb{X}}^{(\mathrm{m})} &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \\ \beta^{(\mathrm{m})}(A) &= (P_{1,:})^\top, \end{aligned}$$

$$
\begin{aligned}
\beta^{(\mathrm{m})}(C) &= (P_{2,:})^\top, \\
\beta^{(\mathrm{m})}(G) &= (P_{3,:})^\top, \\
\beta^{(\mathrm{m})}(T) &= (P_{4,:})^\top.
\end{aligned}
$$

Note that a finite length HMM is represented here with a classical HMM that continuously repeats the finite length sequence. After combination with the quasi model of the background, we will be able to get rid of this continuous repetition. The Mealy equivalent of the motif model is given by $\sigma^{(\mathrm{m})} := (\mathbb{X}^{(\mathrm{m})}, \mathbb{Y}, \Pi^{(\mathrm{m})}, \pi^{(\mathrm{m})}(1))$ where

$$\Pi^{(\mathrm{m})}(\mathrm{y}) = \beta(\mathrm{y})\Pi_{\mathbb{X}}, \quad \mathrm{y} \in \{A, C, G, T\}. \tag{7.8}$$

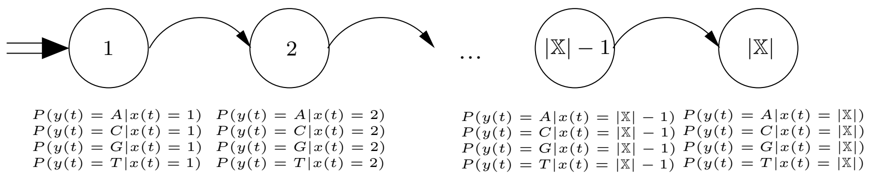

1
2
...
|𝕏| − 1
|𝕏|
$P(y(t) = A|x(t) = 1)$
$P(y(t) = C|x(t) = 1)$
$P(y(t) = G|x(t) = 1)$
$P(y(t) = T|x(t) = 1)$
$P(y(t) = A|x(t) = 2)$
$P(y(t) = C|x(t) = 2)$
$P(y(t) = G|x(t) = 2)$
$P(y(t) = T|x(t) = 2)$
$P(y(t) = A|x(t) = |\mathbb{X}| - 1)$
$P(y(t) = C|x(t) = |\mathbb{X}| - 1)$
$P(y(t) = G|x(t) = |\mathbb{X}| - 1)$
$P(y(t) = T|x(t) = |\mathbb{X}| - 1)$
$P(y(t) = A|x(t) = |\mathbb{X}|)$
$P(y(t) = C|x(t) = |\mathbb{X}|)$
$P(y(t) = G|x(t) = |\mathbb{X}|)$
$P(y(t) = T|x(t) = |\mathbb{X}|)$

Figure 7.5: *We here give an example of a fixed length hidden Markov model. This kind of hidden Markov models is used to model motifs based on their position weight matrix.*

To model the background, one is typically given a long sequence of which it is known that no motifs are present. One can either make a positive hidden Markov model or a quasi hidden Markov model of the background. We explained in the previous parts of this section that the output filtering, estimation, and smoothing methods work both with quasi as well as with positive hidden Markov models. We therefore decide to work with a quasi model of the background, as it gives a better model for the same system order.

A quasi model of the background is given by $\sigma^{(\mathrm{b})} := (\mathbb{Q}^{(\mathrm{b})}, \mathbb{Y}, A^{(\mathrm{b})}, c^{(\mathrm{b})}, b^{(\mathrm{b})})$. It can be obtained by estimating the string probabilities of strings up to a certain length from the given background DNA string and subsequently applying an approximate realization method of Section 4.3.3. We suppose without loss of generality that $b^{(\mathrm{b})} = e$.

Finally, denote the probability to go from background to motif by $\Pi_{\mathrm{b,m}}$, and the probability to go from background to background by $\Pi_{\mathrm{b,b}} = 1 - \Pi_{\mathrm{b,m}}$.

Now the switched quasi HMM consisting of the individual HMMs $\sigma^{(\mathrm{m})}$ and $\sigma^{(\mathrm{b})}$ is given by $(\mathbb{Y} \times \mathbb{Z}, \mathbb{X}^{(\mathrm{m})} \cup \mathbb{X}^{(\mathrm{b})}, A^{(\mathrm{m,b})}, c^{(\mathrm{m,b})}, e)$ where $\mathbb{Z} = \{\mathrm{m}, \mathrm{b}\}$ and

$$
\begin{aligned}
A^{(\mathrm{m,b})}(\mathrm{y}, \mathrm{m}) &= \left[\begin{array}{c|c} \Pi^{(\mathrm{m})}_{1:|\mathbb{X}^{(\mathrm{m})}|-1,:}(\mathrm{y}) & \mathbf{0} \\ \mathbf{0} & \Pi^{(\mathrm{m})}_{|\mathbb{X}^{(\mathrm{m})}|,1}(\mathrm{y})c^{(\mathrm{b})} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right], \quad \forall \mathrm{y} \in \mathbb{Y}, \\
A^{(\mathrm{m,b})}(\mathrm{y}, \mathrm{b}) &= \left[\begin{array}{cc|c} \multicolumn{2}{c|}{\mathbf{0}} & \mathbf{0} \\ \hline \Pi_{\mathrm{b,m}}A^{(\mathrm{b})}(\mathrm{y})e & \mathbf{0} & \Pi_{\mathrm{b,b}}A^{(\mathrm{b})}(\mathrm{y}) \end{array}\right], \quad \forall \mathrm{y} \in \mathbb{Y},
\end{aligned} \tag{7.9}
$$

$$c^{(\mathrm{m,b})} \;=\; \left[\begin{array}{cc|c} \Pi_{\mathrm{b,m}} & \mathbf{0} & \Pi_{\mathrm{b,b}} c^{\mathrm{b}} \end{array}\right].$$

We now apply the described method to the problem of detecting known motifs in human muscle-specific genes [116]. The data for this problem is available at `http://www.stat.ucla.edu/~zhou/CisModule/` [119] and consists of

- 29 sequences $\mathbf{y}_1, \ldots \mathbf{y}_{29}$ of length 200 with instances of motifs Mef-2 (length 12), Myf (length 12), Sp-1 (length 11), SRF (length 13) and TEF (length 12),
- 40 background $\mathbf{u}_1, \ldots \mathbf{u}_{40}$ sequences of length 200 without motifs,
- PWM models of the motifs Mef-2, Myf, Sp-1, SRF and TEF.

The goal is to find the location of motif instances of the 5 different types in the sequences $\mathbf{y}_1, \ldots \mathbf{y}_{29}$.

$$P^{(\mathrm{Mef-2})} = \begin{bmatrix} 0.4525 & 0.0045 & 0.4072 & 0.1357 \\ 0.0045 & 0.0901 & 0.9009 & 0.0045 \\ 0.0045 & 0.5430 & 0.0905 & 0.3620 \\ 0.0045 & 0.0045 & 0.0045 & 0.9865 \\ 0.9865 & 0.0045 & 0.0045 & 0.0045 \\ 0.0045 & 0.0045 & 0.0045 & 0.9865 \\ 0.2703 & 0.0045 & 0.0045 & 0.7207 \\ 0.0901 & 0.0045 & 0.0045 & 0.9009 \\ 0.1351 & 0.0045 & 0.0045 & 0.8559 \\ 0.1802 & 0.0045 & 0.0045 & 0.8108 \\ 0.9865 & 0.0045 & 0.0045 & 0.0045 \\ 0.4525 & 0.0045 & 0.4525 & 0.0905 \end{bmatrix}, \quad P^{(\mathrm{Myf})} = \begin{bmatrix} 0.4348 & 0.4969 & 0.0621 & 0.0062 \\ 0.5556 & 0.0062 & 0.4321 & 0.0062 \\ 0.2484 & 0.1242 & 0.6211 & 0.0062 \\ 0.0062 & 0.9259 & 0.0617 & 0.0062 \\ 0.9816 & 0.0061 & 0.0061 & 0.0061 \\ 0.4321 & 0.0062 & 0.5556 & 0.0062 \\ 0.0062 & 0.9259 & 0.0617 & 0.0062 \\ 0.3704 & 0.0062 & 0.0062 & 0.6173 \\ 0.0061 & 0.0061 & 0.9816 & 0.0061 \\ 0.0062 & 0.6173 & 0.3704 & 0.0062 \\ 0.3704 & 0.0062 & 0.0062 & 0.6173 \\ 0.0061 & 0.0061 & 0.9816 & 0.0061 \end{bmatrix},$$

$$P^{(\mathrm{Sp-1})} = \begin{bmatrix} 0.0082 & 0.0820 & 0.9016 & 0.0082 \\ 0.0081 & 0.0081 & 0.9756 & 0.0081 \\ 0.0081 & 0.0081 & 0.9756 & 0.0081 \\ 0.0081 & 0.0081 & 0.9756 & 0.0081 \\ 0.1653 & 0.4959 & 0.0083 & 0.3306 \\ 0.0081 & 0.0081 & 0.9756 & 0.0081 \\ 0.0081 & 0.0081 & 0.9756 & 0.0081 \\ 0.0083 & 0.0826 & 0.8264 & 0.0826 \\ 0.0083 & 0.1653 & 0.6612 & 0.1653 \\ 0.2500 & 0.1667 & 0.3333 & 0.2500 \\ 0.0826 & 0.4132 & 0.4959 & 0.0083 \end{bmatrix}, \quad P^{(\mathrm{SRF})} = \begin{bmatrix} 0.3333 & 0.2381 & 0.3333 & 0.0952 \\ 0.4286 & 0.0952 & 0.1905 & 0.2857 \\ 0.0047 & 0.9859 & 0.0047 & 0.0047 \\ 0.0047 & 0.8019 & 0.0047 & 0.1887 \\ 0.8491 & 0.0047 & 0.0047 & 0.1415 \\ 0.4245 & 0.0047 & 0.0047 & 0.5660 \\ 0.7143 & 0.0476 & 0.1905 & 0.0476 \\ 0.3791 & 0.0047 & 0.0948 & 0.5213 \\ 0.9859 & 0.0047 & 0.0047 & 0.0047 \\ 0.6635 & 0.0948 & 0.0047 & 0.2370 \\ 0.0047 & 0.0047 & 0.9859 & 0.0047 \\ 0.0047 & 0.0047 & 0.9859 & 0.0047 \\ 0.3318 & 0.3791 & 0.2844 & 0.0047 \end{bmatrix},$$

$$P^{(\mathrm{TEF})} = \begin{bmatrix} 0.0833 & 0.5000 & 0.0833 & 0.3333 \\ 0.7377 & 0.0082 & 0.2459 & 0.0082 \\ 0.0081 & 0.9756 & 0.0081 & 0.0081 \\ 0.9756 & 0.0081 & 0.0081 & 0.0081 \\ 0.0081 & 0.0081 & 0.0081 & 0.9756 \\ 0.0081 & 0.0081 & 0.0081 & 0.9756 \\ 0.0081 & 0.9756 & 0.0081 & 0.0081 \\ 0.0082 & 0.9016 & 0.0082 & 0.0820 \\ 0.4098 & 0.0082 & 0.0082 & 0.5738 \\ 0.0826 & 0.5785 & 0.3306 & 0.0083 \\ 0.1667 & 0.3333 & 0.2500 & 0.2500 \\ 0.0083 & 0.1653 & 0.6612 & 0.1653 \end{bmatrix}.$$

A motif HMM representation $\sigma^{(\mathrm{m})} = (\mathbb{X}^{(\mathrm{m})}, \mathbb{Y}, \Pi_{\mathbb{X}}^{(\mathrm{m})}, \beta^{(\mathrm{m})}, \pi^{(\mathrm{m})}(1))$ is built for the 5 different motifs using Equation (7.8).

In Section 4.4 we already explained how to build a quasi model $\sigma^{(\mathrm{b})} = (\mathbb{Q}^{(\mathrm{b})}, \mathbb{Y}, A^{(\mathrm{b})}, c^{(\mathrm{b})}, b^{(\mathrm{b})})$ of order 4 of the background sequences.

The probability to go from background to motif is taken equal to $\Pi_{\mathrm{b,m}} = 0.02$. Now, the switched quasi HMMs for the different motifs combined with the background, calculated using Equation (7.9), are given by $\sigma^{(\mathtt{m},\mathrm{b})} = (\mathbb{Y} \times \mathbb{Z}, \mathbb{X}^{(\mathtt{m})} \cup \mathbb{X}^{(\mathrm{b})}, A^{(\mathtt{m},\mathrm{b})}, c^{(\mathtt{m},\mathrm{b})}, e)$, where $\mathbb{Z} = \{\mathtt{m}, \mathrm{b}\}$ and $\mathtt{m} = \mathrm{Mef}-2, \mathrm{Myf}, \mathrm{Sp}-1, \mathrm{SRF}, \mathrm{TEF}$.

For the estimation step, we use the recursive fixed lag smoothing procedure with N equal to the length of the motif. As explained in Section 7.3.2, it suffices for this method to have a quasi HMM instead of a positive HMM. The method calculates for a given output sequence $\mathbf{y}$, using the switched quasi HMM $\sigma^{(\mathtt{m},\mathtt{b})}$ of the background and a motif $\mathtt{m}$, the quantities $\omega(t-N;y_1,y_2,...,y_t)(\mathtt{m})$ for $t=N+1,\ldots,200$. In Figures 7.6, 7.7 and 7.8, we show the results for $\mathbf{y}=\mathbf{y}_2,\mathbf{y}_{11},\mathbf{y}_{27}$ and $\mathtt{m}=\mathrm{Mef}-2,\mathrm{Myf},\mathrm{Sp}-1,\mathrm{SRF},\mathrm{TEF}$.

The results are compared to the results obtained using *Motifscanner* [2] applied with a third order background model and default parameters except for "strand = single (s=0)". It is clear that all motifs found by Motifscanner are also detected by our method. However, our method calculates a probability of occurence for each motif over the complete sequence. Hence additional regions where motifs might be present are detected. It might be interesting to investigate these regions biologically. Another advantage is that our method provides a way to determine the needed order of the background model and allows that the background model is a quasi model instead of a positive hidden Markov model.

We conclude that the fixed-lag smoothing method is well suited to detect known motifs in nucleotide sequences. The method works with a quasi model of the background. There are many advantages of working with quasi models. First of all there is no need to calculate a positive HMM representation of the background, which is a computational expensive task. Second, the order of a quasi realization is typically lower than the order of a positive realization which makes the estimation less expensive. In addition, by looking at the singular values of the hankel matrix, the needed order of the quasi model can be determined.

7.8 The linear stochastic case

In this section we consider state filtering for linear stochastic models. We show that also for this problem it suffices to have a quasi model. This is in analogy with the output filtering problem for hidden Markov models. We do not go into detail about prediction and smoothing for linear stochastic models.

First define

$$\hat{x}(t) := E(x(t)|y(1)=y_1,y(2)=y_2,...,y(t-1)=y_{t-1}).$$

Now the Kalman filter algorithm below provides a solution to the recursive state filtering problem for linear stochastic models.

Algorithm 7.7. *Given a linear stochastic model (A,C,P,Q,R,S), then the following equations define a recursive algorithm to compute $\hat{x}(t)$*

$$\hat{x}(t) = A\hat{x}(t-1) + K(y(t-1) - C\hat{x}(t-1)), \tag{7.10}$$

with

$$\begin{aligned} K &= (A\tilde{P}C^\top + S)(C\tilde{P}C^\top + R)^{-1}, \\ \tilde{P} &= A\tilde{P}A^\top + Q - (A\tilde{P}C^\top + S)(C\tilde{P}C^\top + R)^{-1}(A\tilde{P}C^\top + S)^\top, \end{aligned}$$

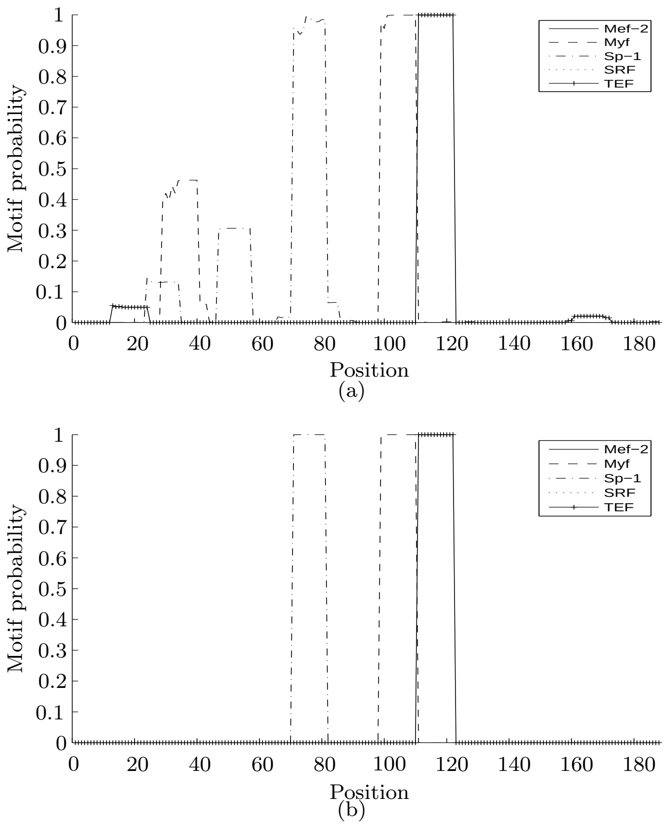


Figure 7.6: *The probability* $\omega(t-N; y_1, y_2, ..., y_t)(m)$ *for* $t = N+1, \ldots, 200$ *is shown for* $\mathbf{y} = \mathbf{y}_2$ *and* $m = \text{Mef}-2, \text{Myf}, \text{Sp}-1, \text{SRF}, \text{TEF}$ *(Subfigure (a)). The results are compared with the method proposed in [2] (Subfigure (b)).*

where $\tilde{P}$ *is the error covariance matrix defined as*

$$\tilde{P} = E[(x(t) - \hat{x}(t))(x(t) - \hat{x}(t))^\top].$$

The matrix K is called the *Kalman gain.* It can be shown that the Kalman gain can be calculated in an alternative way using

$$\begin{aligned} K &= (G - A\hat{P}C^\top)(\Lambda(0) - C\hat{P}C^\top)^{-1}, \\ \hat{P} &= A\hat{P}A^\top + (G - A\hat{P}C^\top)(\Lambda(0) - C\hat{P}C^\top)^{-1}(G - A\hat{P}C^\top)^\top, \end{aligned}$$

where $\hat{P} = E[\hat{x}(t)\hat{x}(t)^\top]$.

This last form depends only on A, C, G and $\Lambda(0)$. This indicates that the Kalman filter works with a quasi model $(A, C, P^{(q)}, Q^{(q)}, R^{(q)}, S^{(q)})$ where $P^{(q)}$, $Q^{(q)}$ and $R^{(q)}$ are not necessarily positive definite as well as with a

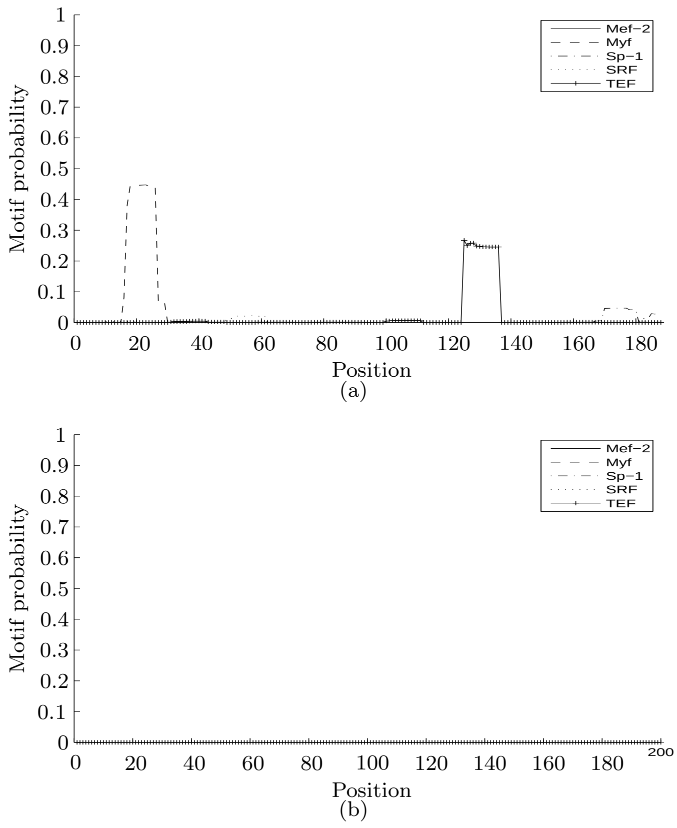


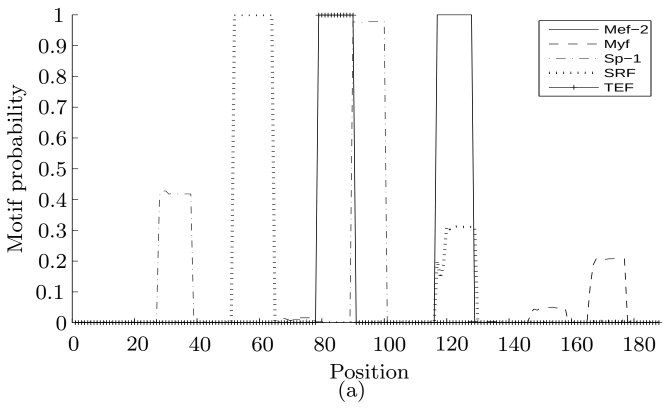
Figure 7.7: *The probability* $\omega(t-N; y_1, y_2, ..., y_t)(m)$ *for* $t = N+1, \ldots, 200$ *is shown for* $\mathbf{y} = \mathbf{y}_{11}$ *and* $m = \text{Mef}-2, \text{Myf}, \text{Sp}-1, \text{SRF}, \text{TEF}$ *(Subfigure (a)). The results are compared with the method proposed in [2] (Subfigure (b)).*

linear stochastic model where P, Q and R are positive definite. Note that this observation is analogous to the observation that for output filtering of hidden Markov models, it suffices to have a quasi realization instead of a positive realization.

7.9 Conclusions

In this chapter we considered estimation problems for hidden Markov models.

First, the recursive state filtering and prediction problem are reviewed. Subsequently, it is shown that for the output filtering and prediction problem, it suffices to have a quasi hidden Markov model instead of a positive hidden Markov model.



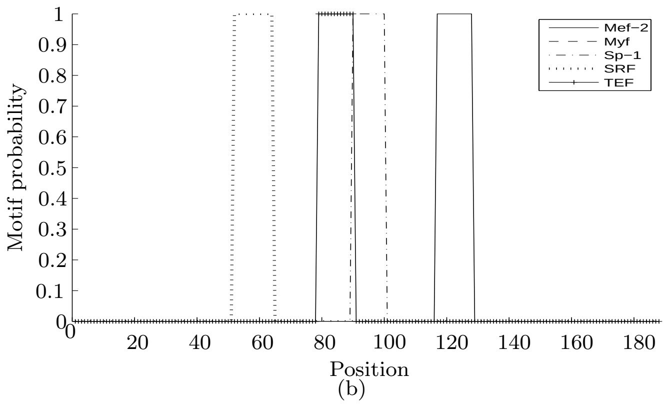


Figure 7.8: *The probability* $\omega(t-N; y_1, y_2, ..., y_t)(m)$ *for* $t = N+1, \ldots, 200$ *is shown for* $\mathbf{y} = \mathbf{y}_{27}$ *and* $m = \text{Mef}-2, \text{Myf}, \text{Sp}-1, \text{SRF}, \text{TEF}$ *(Subfigure (a)). The results are compared with the method proposed in [2] (Subfigure (b)).*

Subsequently, the recursive fixed-point and fixed-lag state smoothing problem are introduced and solved. Again, we prove that for the fixed-point and fixed-lag output smoothing problem, it suffices to have a quasi hidden Markov model instead of a positive hidden Markov model.

Next, the fixed interval state smoothing problem using the Viterbi algorithm is reviewed. The method is adapted to solve the fixed-interval output smoothing problem for hidden Markov models.

Finally, switched hidden Markov models were defined and a method was presented to determine the operation mode of a switched hidden Markov model. This method is succesfully applied on the problem of finding motifs in DNA sequences.

Chapter 8

Conclusions and directions for further research

This chapter summarizes the most important results obtained in this thesis and suggests some directions for further research.

8.1 Conclusions

Hidden Markov models are models with a finite-valued state space that are used to model finite-valued output processes. Linear stochastic models on the other hand have a finite-dimensional state space and are used to model finite-dimensional output processes. Conceptually, there is a close link between hidden Markov and linear stochastic models. However, many theoretical questions that are "solved" for linear stochastic models (realization, identification, filtering,...), are still open for hidden Markov models. These theoretical questions are the first main objective of this thesis.

The solution to most of the theoretical questions concerning linear stochastic models makes use of the singular value decomposition. To solve the corresponding problems for hidden Markov models, there is need for the nonnegative matrix factorization as well as variants to this factorization. The derivation of variants on the nonnegative matrix factorization is the second objective of this thesis. We aim at keeping the variants on the nonnegative matrix factorization as general as possible such that they are not only useful in the domain of hidden Markov models, but can be used in other applications too.

We now summarize our contributions to matrix factorizations (Section 8.1.1) and hidden Markov models (Section 8.1.2).

8.1.1 Matrix factorizations

In recent years, the approximate nonnegative matrix factorization problem has gained lots of interest both from theoretical as from algorithmical point of view as well as in applications. It consists of decomposing a nonnegative matrix M into a low rank product VH with V and H nonnegative. We introduce two variants on the nonnegative matrix factorization.

The first variant is the structured nonnegative matrix factorization. It consists of approximating a square nonnegative matrix P with a product $VAV^\top$ where the dimension of A is small. When using the Kullback-Leibler divergence as performance criterion, we prove that an optimal approximation $VAV^\top$ has the same element sum as the original matrix P. Next, we propose update formulas of which we proved that, if they converge, they converge to a stationary point of the divergence. We show that the structured nonnegative matrix approximation can be used to cluster points based on their distance matrix.

The second introduced variant is the nonnegative matrix factorization without nonnegativity constraints on the factors. The purpose there is to decompose a matrix M into a low rank product VH where V and H can contain negative values, but the approximation VH is constrained to be nonnegative. By allowing the factors V and H to contain negative elements a better approximation can be obtained with the same inner dimension compared to the case where the factors V and H are constrained to be nonnegative. We apply this decomposition to the problem of compressing a database containing facial images. It turns out that the proposed method outperforms existing methods.

8.1.2 Hidden Markov models

Severeral types of hidden Markov models have been considered in the literature: quasi Mealy models, positive Mealy models and positive Moore models. In this thesis, the importance of quasi Mealy models is explained. For instance, several estimation problems for hidden Markov models can be solved using quasi HMMs (see Chapter 7). Nevertheless, for some applications a quasi model is not sufficient and it is important to have a positive model instead of a quasi model. For that reason, we concentrated both on methods to obtain quasi models from data as well as on methods to obtain positive models from data.

Quasi-realization for hidden Markov models

In the literature only the exact quasi realization problem for hidden Markov models has been considered. In this thesis, we consider the partial quasi realization problem and the approximate partial quasi realization problem for hidden Markov models. These problems are more important from practical point of view. Concerning the partial realization problem, we prove that, under a certain rank condition, the problem can be solved using the same algorithm as for the quasi realization problem. In addition it is proven that, again under the rank condition, the obtained model is the unique minimal solution to the partial quasi realization problem. Subsequently, we provide methods to solve the approximate partial realization problem. The first methods aim at finding a low rank approximation of the generalized Hankel matrix containing the string probabilities and subsequently apply the quasi realization algorithm. The last method builds a full-order balanced model and then reduces the model to obtain a low order approximate model. The approximate quasi realization procedure is applied to the modeling of DNA sequences.

Realization for hidden Markov models

We considered the approximate partial realization problem for positive hidden Markov models. We prove that the approximate Moore realization problem for string probabilities of strings up to length two can be solved using the structured nonnegative matrix factorization. By generalizing this method, we obtain a method to solve the approximate partial Mealy realization problem for string probabilities of strings up to arbitrary length t.

We also consider the equivalence problem for positive Moore and Mealy hidden Markov models. We show that, under certain conditions, a positive Moore model has only trivial equivalents obtained by permuting the states. In case these conditions do no hold there exists a whole class of equivalent models. We provide a procedure to check whether two Moore models are equivalent and give a description of the complete set of equivalent Moore models. We also provide a test to check whether two Mealy models are equivalent and give a description of the complete set of equivalent Mealy models. We have applied the approximate realization procedure to the modeling of DNA sequences.

Identification for hidden Markov models

We provide a new identification procedure for hidden Markov models. Classically, identification is solved using Baum-Welch, a maximum likelihood approach. We propose a method that is inspired by subspace identification for linear stochastic models. For a given model order, the method first estimates the state sequence directly from the output data. Subsequently, the system matrices are estimated from the obtained state sequence and the given output sequence. In a simulation example, we show that the method outperforms the Baum-Welch algorithm. The subspace inspired identification method has been used to model sequences from the HIV genome.

Estimation for hidden Markov models

We provide a method to solve the recursive fixed-point and fixed-lag state and output smoothing problem. We show that for the output filtering, output prediction, fixed-point and fixed-lag output smoothing problem, it suffices to have a quasi hidden Markov model instead of a positive hidden Markov model. We propose a technique for determining the operation mode of a switched hidden Markov model and apply this technique to the problem of finding motifs in DNA sequences.

8.2 Directions for further research

In this section we give some directions for further research. In Section 8.2.2 we consider directions concerning matrix factorizations and in Section 8.2.2 concerning hidden Markov models.

8.2.1 Matrix factorizations

We here discuss some open question concerning nonnegative matrix factorization techniques.

- The singular value decomposition has a *nesting property.* This means that the optimal rank l SVD-truncation of the optimal rank k SVD-truncation (with $l < k$) of a matrix M is equal to the optimal rank l SVD-truncation of the matrix M. It would be interesting to have a nonnegative matrix factorization with a similar type of nesting property.

- The nonnegative matrix factorization without nonnegativity constraints on the factors, proposed in this thesis, elegantly allows to deal with upper as well as lower bounds (nonnegativity) on the elements of the approximation. It would be interesting to derive update formulas for all types of nonnegative matrix factorization problems where upper bounds can be imposed next to lower bounds.

8.2.2 Hidden Markov models

We here discuss some open question concerning hidden Markov models.

- In the approximate positive realization problem for hidden Markov models one could use a weighted Kullback-Leibler divergence (as defined in [21]) instead of a uniform divergence and try to define update formulas using this new distance measure. The following applications illustrates the use of this kind of realization algorithms. Suppose the output symbols of the HMM are measured in the presence of noise, which means that certains symbols can be flipped. In that case some output string probabilities can be determined more precisely than others. In that case a weighted realization approach can be helpfull.

- The derivation of necessary and sufficient conditions for string probabilities to be representable by a positive Mealy HMM is still an interesting open research problem. In addition, it would be nice to determine beforehand the order of a minimal positive realization of the string probabilities. It is also an open problem to derive an exact minimal positive realization algorithm for hidden Markov models.

- All methods to obtain a model developed in this thesis (quasi realization, positive realization and identification) build a model in which no prior knowledge is incorparated. However, in some practical applications, prior knowledge on the structure of the underlying Markov chain is available. For example a linear structured hidden Markov chain (see Figure 8.1), or a hidden Markov model in which some transitions are not allowed. It would be interesting to have realization and identification methods that allow to deal with this kind of prior knowledge. Incorporating prior knowledge will allow to obtain even better models.

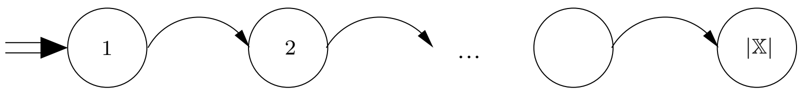


Figure 8.1: *In some practical applications of hidden Markov models, prior knowledge on the structure of the underlying Markov chain is available. We here give an example of a linear structured Markov chain. It would be interesting to have modeling methods that allow to deal with this kind of prior knowledge.*

- It is an interesting open problem to check whether it is possible to define canonical forms of hidden Markov models and whether these forms can help in developping better identification procedures. For linear stochastic models several canonical forms do exist (forward innovation form, backward innovation form) and these forms do have importance in several identification procedures.
- An interesting open problem is model reduction for hidden Markov models: given a high order hidden Markov model, find a hidden Markov model of a given order that approximates the high order model optimally in a certain sense. In [66], a model reduction method for quasi hidden Markov models is proposed. Using a heuristic trick this model reduction method is also used for positive hidden Markov models. It would be interesting to investigate the model reduction problem into more detail.
- In this thesis, we considered hidden Markov models without external inputs. It would be interesting to define hidden Markov models with an input that can be controlled by the user. A possible description of a Mealy model could be $(\mathbb{X}, \mathbb{Y}, \mathbb{U}, \Pi, \pi(1))$, where $\mathbb{X}$ and $\mathbb{Y}$ are the state and output alphabeth as before and $\mathbb{U}$ is the input alphabeth with cardinality $|\mathbb{U}|$. Π is a mapping from $\mathbb{Y} \times \mathbb{U}$ to $\mathbb{R}_+^{|\mathbb{X}| \times |\mathbb{X}|}$ defined as

  $$\Pi_{ij}(\mathtt{y}, \mathtt{u}) = P(y(t) = \mathtt{y}, x(t+1) = j | x(t) = i, u(t) = \mathtt{u})$$

  and $\pi(1)$ is defined as before as $\pi_i(1) = P(x(1) = i)$. It could be investigated whether the developped quasi realization theory, realization theory, identification and estimation methods could be extended to hidden Markov models with inputs.

- In control problems one is given a desired output sequence and the problem is to design the input sequence such that the actual output approximates the desired output optimally in a to be defined sense. To the best of our knowledge, the control problem for hidden Markov models has not been considered yet. In order to be able to say whether an output sequence approximates the desired output better than onother output sequence, we need to define a distance measure between finite-valued output sequences. If a distance measure $d(\mathrm{y}^{(1)}, \mathrm{y}^{(2)})$ between the output symbols $\mathrm{y}^{(1)}$ and $\mathrm{y}^{(2)}$ is given, the distance between output sequences $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$ of length T can be defined as

$$D(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = \sum_{i=1}^{T} d(y_i^{(1)}, y_i^{(2)}).$$

Appendix A

Proof of Theorem 2.4

To prove that M_k is the unique optimal approximation of M in the Frobenius norm, we suppose that M'_k is another optimal approximation and want to prove that $M'_k = M_k$. Let

$$M'_k = U' \begin{bmatrix} \Sigma'_k & 0 \\ 0 & 0 \end{bmatrix} V'^\top$$

with $\Sigma'_k \in \mathbb{R}^{k\times k}$, be an SVD of M'_k. Then $\begin{bmatrix} \Sigma'_k & 0 \\ 0 & 0 \end{bmatrix}$ is an optimal rank k approximation in the Frobenius norm of $N := (U')^\top M V'$. Partition

$$N = \begin{bmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{bmatrix}$$

conformal with the partition $\begin{bmatrix} \Sigma'_k & 0 \\ 0 & 0 \end{bmatrix}$.

Observe that, since

$$\text{rank}(\begin{bmatrix} \Sigma'_k & N_{12} \\ 0 & 0 \end{bmatrix} \leq k$$

and

$$[N_{12} \neq 0] \Rightarrow [||N - \begin{bmatrix} \Sigma'_k & N_{12} \\ 0 & 0 \end{bmatrix} ||_F < ||N - \begin{bmatrix} \Sigma'_k & 0 \\ 0 & 0 \end{bmatrix} ||_F],$$

we obtain $N_{12} = 0$. Similarly, $N_{21} = 0$. Therefore $N = \begin{bmatrix} N_{11} & 0 \\ 0 & N_{22} \end{bmatrix}$. Observe also that, since

$$\text{rank}(\begin{bmatrix} \Sigma'_k - N_{11} & 0 \\ 0 & 0 \end{bmatrix} \leq k$$

and

$$[N_{11} \neq \Sigma'_k] \Rightarrow [||N - \begin{bmatrix} N_{11} & 0 \\ 0 & 0 \end{bmatrix} ||_F < ||N - \begin{bmatrix} \Sigma'_k & 0 \\ 0 & 0 \end{bmatrix} ||_F],$$

we obtain $N_{11} = \Sigma'_k$. Therefore $N = \begin{bmatrix} \Sigma'_k & 0 \\ 0 & N_{22} \end{bmatrix}$. Next, let $N_{22} = U_{22}\Sigma''_k V_{22}^\top$ be an SVD of N_{22}, and note that

$$N' := \begin{bmatrix} I & 0 \\ 0 & U_{22}^\top \end{bmatrix} N \begin{bmatrix} I & 0 \\ 0 & V_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & U_{22}^\top \end{bmatrix} (U')^\top M V' \begin{bmatrix} I & 0 \\ 0 & V_{22} \end{bmatrix}$$

is diagonal: $N' = \begin{bmatrix} \Sigma'_k & 0 \\ 0 & \Sigma''_k \end{bmatrix}$, and has $\begin{bmatrix} \Sigma'_k & 0 \\ 0 & 0 \end{bmatrix}$ as an optimal rank k approximation. This obviously implies that the smallest diagonal element of Σ'_k is larger than the largest diagonal element of Σ''_k. It follows that

$$M = U' \begin{bmatrix} I & 0 \\ 0 & U_{22} \end{bmatrix} \begin{bmatrix} \Sigma'_k & 0 \\ 0 & \Sigma''_k \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & V_{22}^\top \end{bmatrix} V'^\top$$

is an SVD of M and that

$$M'_k = U' \begin{bmatrix} \Sigma'_k & 0 \\ 0 & 0 \end{bmatrix} V'^\top = U' \begin{bmatrix} I & 0 \\ 0 & U_{22} \end{bmatrix} \begin{bmatrix} \Sigma'_k & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & V_{22}^\top \end{bmatrix} V'^\top$$

is a rank k SVD-truncation of M.

Now, if the gap condition $\sigma k(M) > \sigma_{k+1}(M)$ holds, then the rank k SVD-truncation is unique. Hence $M'_k = M_k$. Conclude that M_k is then the *unique* optimal rank k approximation in the Frobenius norm of M.

Appendix B

Semialgebraic sets - Tarski-Seidenberg quantifier elimination

In this appendix, we summarize the principle of Tarski-Seidenberg and the relation to semialgebraic sets. This summary is based on [22].

A *semialgebraic subset* of $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ satisfying a boolean combination of polynomial equations and inequalities with real coefficients. In other words, the semialgebraic subsets of $\mathbb{R}^n$ form the smallest class $\mathcal{SA}_n$ of subsets of $\mathbb{R}^n$ such that:

- If g is a polynomial in n variables, then $\{x \in \mathbb{R}^n : g(x) = 0\} \in \mathcal{SA}_n$ and $\{x \in \mathbb{R}^n : g(x) > 0\} \in \mathcal{SA}_n$.
- If $A \in \mathcal{SA}_n$ and $B \in \mathcal{SA}_n$, then $A \cup B$, $A \cap B$ and $A \setminus B$ are in $\mathcal{SA}_n$.

As a consequence of the Tarski-Seidenberg principle [91, 97], the class of semialgebraic sets is closed under projection.

Theorem B.1. *Let A be a semialgebraic subset of $\mathbb{R}^n$ and $P : \mathbb{R}^n \mapsto \mathbb{R}^p$, the projection on the first p coordinates. Then $P(A)$ is a semialgebraic subset of $\mathbb{R}^p$.*

Now consider a first order formula over the reals having the form

$$(Q_1 x^{(1)} \in \mathbb{R}^{n_1}) \dots (Q_l x^{(l)} \in \mathbb{R}^{n_l}) P(y, x^{(1)}, \dots, x^{(l)}), \tag{B.1}$$

where $Q_\lambda, \lambda = 1, \dots, l$ is a quantifier: either $\exists$ ("there exists") or $\forall$ ("for all"), where $y = [y_1, \dots, y_{n_0}]^\top$ are free variables and where $P(y, x^{(1)}, \dots, x^{(l)})$ is a quantifier-free Boolean formula, i.e. a combination of atomic predicates. The atomic predicates are supposed to be of the form $g_\kappa(y, x^{(1)}, \dots, x^{(l)}) \Delta_\kappa 0, \kappa = 1, \dots, k$, where $g_\kappa : \prod_{\lambda=0}^{l} \mathbb{R}^{n_\lambda} \mapsto \mathbb{R}$ is a polynomial of degree at most $d \geq 2$ and

Δ_i is one of the following relations $\geq$, $>$, $=$, $\neq$, $\leq$ and $<$. $P(y, x^{(1)}, \ldots, x^{(l)})$ is determined by a Boolean function $\mathbb{P} : \{0,1\}^k \mapsto \{0,1\}$ and a function $B : \prod_{\lambda=0}^{l} \mathbb{R}^{n_\lambda} \mapsto \{0,1\}^k$, where $P := \mathbb{P} \circ B$, and for $\kappa = 1, \ldots, k$

$$B(y, x^{(1)}, \ldots, x^{(l)})_\kappa = \begin{cases} 1 & \text{if } g_\kappa(y, x^{(1)}, \ldots, x^{(l)}) \Delta_\kappa 0, \\ 0 & \text{otherwise.} \end{cases}$$

It is clear that the solution set in $\mathbb{R}^{n_0}$ of (B.1) is a semialgebraic set as it is the projection of a semialgebraic set in $\prod_{\lambda=0}^{l} \mathbb{R}^{n_\lambda}$. It can now be shown [88, 91, 97], that the semialgebraic set can be *constructed*, i.e. the first order formula (B.1) can be written in an equivalent form without quantifiers. The operations that are needed to eliminate the quantifiers are restricted to additions, subtractions, multiplications, divisions, comparisions and the evaluation of Boolean functions. In [88], an algorithm for quantifier elimination is described that requires at most $(kd)^{2^{O(l)} \prod_\lambda n_\lambda}$ multiplications and additions, and at most $(kd)^{O(\sum_\lambda n_\lambda)}$ calls to $\mathbb{P}$. The method requires no divisions. The quantifier elimination algorithm constructs a quantifier-free formula of the following form

$$\bigvee_{\mu=1}^{m} \bigwedge_{\nu=1}^{n_\mu} h_{\mu\nu}(y) \Delta_{\mu\nu} 0,$$

where $m \leq (kd)^{2^{O(l)} \prod_\lambda n_\lambda}$, where $n_\mu \leq (kd)^{2^{O(l)} \prod_\lambda n_\lambda}$, for $\mu = 1, \ldots m$, where the degree of each of the polynomials $h_{\mu\nu}$ is at most $(kd)^{2^{O(l)} \prod_\lambda n_\lambda}$ and where each $\Delta_{\mu\nu}$ is one of the following relations $\geq$, $>$, $=$, $\neq$, $\leq$ and $<$.

Appendix C

The Expectation-Maximization algorithm

The Expectation-Maximization algorithm [32] (EM algorithm) is an iterative procedure to compute the Maximum Likelihood (ML) estimate in the presence of missing or hidden data. We here give a short review of the algorithm based on the tutorial [23]. In ML estimation, we wish to estimate the model parameter for which the observed data are most likely. Each iteration of the EM algorithm consists of two steps: The Expectation step and the Maximization step. In the Expectation step, the missing data are estimated given the observed data and current estimate of the model parameters. This is achieved using the conditional expectation, explaining the choice of terminology. In the Maximization step, the likelihood function is maximized under the assumption that the missing data are known. The estimate of the missing data from the Expectation step are used instead of the missing data. The algorithm is guaranteed to increase the likelihood at each iteration.

Let $\mathbf{y}$ be random vector which results from a parametrized family. We wish to find λ such that $P(\mathbf{y}|\lambda)$ is a maximum. This is known as the maximum likelihood estimate for λ. In order to estimate λ, it is typical to introduce the *log likelihood function* defined as

$$L(\lambda) = \log P(\mathbf{y}|\lambda).$$

Since log is a strictly increasing function, the value of λ which maximizes $P(\mathbf{y}|\lambda)$ also maximizes $L(\lambda)$. The EM algorithm is an iterative procedure for maximizing $L(\lambda)$. Assume that after the t-th iteration, the current estimate for λ is given by $\lambda^{(t)}$. Since the objective is to maximize $L(\lambda)$, we wish to compute an updated estimate λ such that

$$L(\lambda) > L(\lambda^{(t)})$$

Equivalently, we want to maximize the difference

$$L(\lambda) - L(\lambda^{(t)}) = \log P(\mathbf{y}|\lambda) - \log P(\mathbf{y}|\lambda^{(t)}). \tag{C.1}$$

By introducing hidden variables $\mathbf{x}$, Equation (C.1) becomes

$$L(\lambda) - L(\lambda^{(t)}) = \log \sum_{\mathbf{x}} P(\mathbf{y}|\mathbf{x}, \lambda) P(\mathbf{x}|\lambda) - \log P(\mathbf{y}|\lambda^{(t)}).$$

Now using Jensen's inequality, it can be shown that

$$L(\lambda) - L(\lambda^{(t)}) \geq \Delta(\lambda|\lambda^{(t)})$$

where

$$\Delta(\lambda|\lambda^{(t)}) := \sum_{\mathbf{x}} P(\mathbf{x}|\mathbf{y}, \lambda^{(t)}) \log \frac{P(\mathbf{y}|\mathbf{x}, \lambda) P(\mathbf{x}|\lambda)}{P(\mathbf{x}|\mathbf{y}, \lambda^{(t)}) P(\mathbf{y}|\lambda^{(t)})}.$$

Now define $l(\lambda|\lambda^{(t)}) := L(\lambda^{(t)}) + \Delta(\lambda|\lambda^{(t)})$, such that

$$L(\lambda) \geq l(\lambda|\lambda^{(t)}).$$

So $l(\lambda|\lambda^{(t)})$ is bounded above by the likelihood function $L(\lambda)$. Additionally, it can be shown that $l(\lambda|\lambda^{(t)}) = L(\lambda^{(t)})$.

The objective is to chose a value of λ such that $L(\lambda)$ is maximized. We have shown that the function $l(\lambda|\lambda^{(t)})$ is bounded above by the likelihood function $L(\lambda)$ and that the value of the functions $l(\lambda|\lambda^{(t)})$ and $L(\lambda)$ are equal at the current estimate for $\lambda = \lambda^{(t)}$. Therefore, any λ which increases $l(\lambda|\lambda^{(t)})$ in turn increases $L(\lambda)$. In order to achieve the greatest possible increase in the value of $L(\lambda)$, the EM algorithm selects λ such that $l(\lambda|\lambda^{(t)})$ is maximized. We denote the updated value as $\lambda^{(t+1)}$. This process is illustrated in Figure C.1.

Formally, we have after some calculation

$$\begin{aligned} \lambda^{(t+1)} &:= \operatorname*{argmax}_{\lambda} l(\lambda|\lambda^{(t)}) \\ &= \operatorname*{argmax}_{\lambda} Q(\lambda|\lambda^{(t)}), \end{aligned} \tag{C.2}$$

where

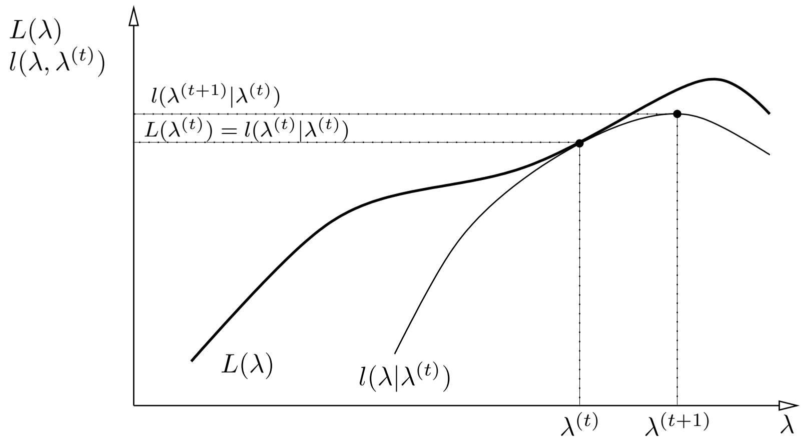
$$Q(\lambda|\lambda^{(t)}) = \sum_{\mathbf{x}} P(\mathbf{x}, \mathbf{y}|\lambda^{(t)}) \log P(\mathbf{x}, \mathbf{y}|\lambda). \tag{C.3}$$

In Equation (C.2) the Expectation and Maximization steps are apparent. The EM algorithm thus consists of iterating the:

1. **E-step**: Determine the conditional expectation

$$\sum_{\mathbf{x}} P(\mathbf{x}, \mathbf{y}|\lambda^{(t)}) \log P(\mathbf{x}, \mathbf{y}|\lambda),$$

2. **M-step**: Maximize this expression with respect to λ.



$L(\lambda)$
$l(\lambda, \lambda^{(t)})$
$l(\lambda^{(t+1)}|\lambda^{(t)})$
$L(\lambda^{(t)}) = l(\lambda^{(t)}|\lambda^{(t)})$
$L(\lambda)$
$l(\lambda|\lambda^{(t)})$
$\lambda^{(t)}$
$\lambda^{(t+1)}$
λ

Figure C.1: *Graphical interpretation of a single iteration of the EM algorithm: The function $l(\lambda|\lambda^{(t)})$ is upper-bounded by the likelihood function $L(\lambda)$. The functions are equal at $\lambda = \lambda^{(t)}$. The EM algorithm chooses $\lambda^{(t+1)}$ as the value of λ for which $l(\lambda|\lambda^{(t)})$ is a maximum. Since $L(\lambda) \geq l(\lambda|\lambda^{(t)})$, increasing $l(\lambda|\lambda^{(t)})$ ensures that the values of the likelihood function $L(\lambda)$ is increased at each step.*

At this point, it is fair to ask what had been gained given that we have simply traded the maximization of $L(\lambda)$ for the maximization of $l(\lambda|\lambda^{(t)})$. The answer lies in the fact that $l(\lambda|\lambda^{(t)})$ takes into account the unobserved or missing data $\mathbf{x}$. In the case where we wish to estimate these variables the EM algorithm provides a framework for doing so. Also, in other situations the maximization of $l(\lambda|\lambda^{(t)})$ is easier as compared with a direct maximization of $L(\lambda)$.

The convergence properties of the EM algorithm are discussed in detail in [76]. Since $\lambda^{(t+1)}$ is chosen to maximize $\Delta(\lambda|\lambda^{(t)})$, we then have that $\Delta(\lambda^{(t+1)}|\lambda^{(t)}) \geq \Delta(\lambda^{(t)}|\lambda^{(t)}) = 0$, so for each iteration the likelihood $L(\lambda)$ is nondecreasing. When the algorithm reaches a fixed point for some $\lambda^{(t)}$ the value $\lambda^{(t)}$ maximizes $l(\lambda)$. Since L and l are equal at $\lambda^{(t)}$, $\lambda^{(t)}$ must be a stationary point of L. The stationary point need not, however, be a local maximum. In [76] it is shown that it is possible for the algorithm to converge to local minima or saddle points in unusual cases.

Bibliography

[1] CBCL Face Database 1. Mit center for biological and computation learning. *http://www.ai.mit.edu/projects/cbcl.*

[2] S. Aerts, G. Thijs, B. Coessens, M. Staes, Y. Moreau, and B. De Moor. Toucan : Deciphering the cis-regulatory logic of coregulated genes. *Nucleic Acids Research*, 31:1753–1764, 2003.

[3] H. Akaike. Stochastic theory of minimal realization. *IEEE Transactions on Automatic Control*, 26:667–673, 1974.

[4] B.D.O. Anderson. The realization problem for hidden Markov models. *Mathematics of Control, Signals, and Systems*, 12:80–120, 1999.

[5] B.D.O. Anderson, M. Deistler, L. Farina, and L. Benvenuti. Nonnegative realization of a system with a nonnegative impulse response. *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, 43:134–142, 1996.

[6] B.D.O. Anderson and J. Moore. *Optimal filtering*. Prentice-Hall, 1979.

[7] S. Andersson. *Hidden Markov models - Traffic modeling and Subspace Methods*. PhD thesis, Lund University, Sweden, 2002.

[8] S. Andersson. Subspace methods for hidden Markov models - algorithms and consistency. Submitted for publication, 2002.

[9] F. Barioli and A. Berman. The maximal cp-rank of rank k completely positive matrices. *Linear Algebra and its Applications*, 363:17–33, 2003.

[10] L. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[11] L. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171, 1970.

[12] L. Benvenuti and L. Farina. A tutorial on the positive realization problem. *IEEE Transactions on Automatic Control*, 49:651–664, 2004.

[13] O. Berg and P. von Hippel. Selection DNA binding sites by regulatory proteins. *Journal of Molecular Biology*, 193:723–750, 1987.

[14] A. Berman and U. Rothblum. A note on the computation of the cp-rank. *Linear Algebra and its Applications*, 419:1–7, 2006.

[15] A. Berman and N. Shaked-Monderer. *Completely Positive Matrices*. World Scientific Publishing Co, New Jersey, 2003.

[16] M. Berry, M. Browne, A. Langville, V. Pauca, and R. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. Submitted for publication, 2006.

[17] J. Berstel and C. Reutenauer. Rational series and their languages. In *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1984.

[18] P. Bickel and Y. Ritov. Inference in hidden markov models I: local asymptotic normality in the stationary case. *Bernoulli*, 2:199–228, 1996.

[19] P. Bickel, Y. Ritov, and T. Ryden. Asymptotic normality of the maximum-likelihood estimator for general hidden Markov models. *Annals of Statistics*, 26:16141635, 1998.

[20] D. Blackwell and L. Koopmans. On the identifiability problem for functions of finite Markov chains. *Annals of Mathematical Statistics*, 28:1011–1015, 1957.

[21] V. Blondel, N. Ho, and P Van Dooren. Algorithms for weighted nonnegative matrix factorizations. Submitted for publication, 2007.

[22] J. Bochnak, M. Coste, and M.F. Roy. Real algebraic geometry, 2nd ed. in English. In *Ergebnisse der Mat.*, volume 36. Springer, Berlin Heidelberg New York, 1998.

[23] S. Borman. The expectation maximization algorithm: a short tutotial. Submitted for publication, 2004.

[24] J. Carlyle. Stochastic finite-state system theory. In L. Zadeh and L. Polak, editors, *Systems Theory*. Mc-Grawhill, New York, 1969.

[25] M. Catral, L. Han, M. Neumann, and R. Plemmons. On reduced rank nonnegative matrix factorization for symmetric nonnegative matrices. *Linear Algebra and its Applications*, 393:107–126, 2004.

[26] M. Chu, R. Funderlic, and R. Plemmons. Structured low rank approximation. *Linear Algebra and its Applications*, 366:157–172, 2003.

[27] J. Coffin. HIV population dynamics in-vivo: Implications for gentic variattion, pathogenesis, and therapy. *Science*, 267:483–489, 1995.

[28] J. Cohen and U. Rothblum. Nonnegative ranks, decompositions and factorizations of nonnegative matrices. *Linear Algebra and its Applications*, 190:149–168, 1993.

[29] J. dan den Hof and J. van Schuppen. Realization of positive linear systems using polyhedral cones. In *Proc. of the 33th IEEE Conference on Decision and Control*, pages 3889–3893, 1994.

[30] P. Davis. *Circulant Matrices.* Chelsea Publishing, New York, 1994.

[31] A. Delcher, D. Harmon, S. Kasif, O. White, and S. Salzberg. Improved microbial gene identification with GLIMMER. *Nucleic Acids Research*, 27:4636–4641, 1999.

[32] A. Dempster, M. Laird, and D. Rubin. Maximum likelihood for incomplete data via the em algoritm. *Journal of the royal statistical society: series B*, 39:1–38, 1977.

[33] J. Van den Hof. *System theory and system identification of compartmental systems.* PhD thesis, Rijksuniversiteit Groningen, 1996.

[34] S. Dharmadhikari. Functions of finite Markov chains. *Annals of Mathematical Statistics*, 34:1022–1032, 1963.

[35] S. Dharmadhikari. Sufficient conditions for a stationary process to be function of a finite Markov chain. *Annals of Mathematical Statistics*, 34:1033–1041, 1963.

[36] S. Dharmadhikari. A characterization of a class of functions of finite Markov chains. *Annals of Mathematical Statistics*, 36:524–528, 1965.

[37] S. Dharmadhikari. Splitting a single state of a stationary process into Markovian states. *Annals of Mathematical Statistics*, 38:1069–1077, 1968.

[38] S. Dharmadhikari. A note on exchangeable processes with states of finite rank. *Annals of Mathematical Statistics*, 40:2207–2208, 1969.

[39] S. Dharmadhikari and M. Nadkarmi. Some regular and non-regular functions of finite Markov chains. *Annals of Mathematical Statistics*, 41:207–213, 1970.

[40] C. Ding, T. Li, and W. Peng. On the equivalence between nonnegative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics and Data Analysis*, 52, 2008.

[41] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *Proc. International Conference on Knowledge Discovery and Data Mining*, 2006.

[42] J. Doob. The elementary gaussian processes. *Annals of Mathematical Statistics*, 15:229–282, 1944.

[43] J. Doob. *Stochastic processes.* John Wiley, New York, 1953.

[44] G. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936.

[45] R. Elliott, L. Aggoun, and J. Moore. *Hidden Markov models: Estimation and Control*. Springer-Verlag, New York, 1995.

[46] P. Faurre. Stochastic realization algorithms. In R. Mehra and D. Lainiotis, editors, *System Identification, Advances and case studies*. Academic Press, 1976.

[47] L. Finesso, A. Grassi, and P. Spreij. Approximation of stationary processes by hidden Markov models. Submitted for publication, 2006.

[48] L. Finesso and P. Spreij. Nonnegative matrix factorization an I-divergence alternating minimization. *Linear Algebra and its Applications*, 416:270–287, 2006.

[49] G. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61:268–278, 1973.

[50] M. Fox and H. Rubin. Functions of processess with Markovian states. *Annals of Mathematical Statistics*, 39:938–946, 1968.

[51] M. Gevers. An innovations approach to least-squares estimation, part vi: Discrete-time innovations representations and recursive estimation. *IEEE Transactions on Automatic Control*, 18:588–600, 1973.

[52] E. Gilbert. The identifiability problem for functions of Markov chains. *Annals of mathematical Statistics*, 30:688–697, 1959.

[53] S. Gillijns. *Kalman filtering techniques for system inversion and data assimilation*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 2007.

[54] I. Gohberg, M. Kaaskoek, and L. Lerer. On minimality in the partial realization problem. *System and Control Letters*, 9:97–104, 1987.

[55] G. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1996.

[56] B. Ho and R. Kalman. Effective construction of linear state-variable models from input-output functions. *Regelungstechnik*, 12:545–548, 1965.

[57] D. Ho and P. van Dooren. Nonnegative matrix factorizations with fixed row and column sums. *Linear Algebra and its Applications*, 2007.

[58] H. Ito, S. Amari, and K. Kobayashi. Identifiabilty of hidden Markov information sources and their minimum degrees of freedom. *IEEE Transaction on Information Theory*, 38:324–333, 1992.

[59] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[60] F. Jelinek. *Statistical methods for speech recognition*. MIT Press, Cambridge, 1997.

[61] R. Kalman. On the general theory of control systems. In *Proc. of the first IFAC Congress*, pages 481–491, 1960.

[62] R. Kalman. Mathematical description of linear dynamical systems. *SIAM Journal of control*, 1:152–192, 1963.

[63] R. Kalman. On minimal partial realizations of a linear input/output map. In R. Kalman and N. De Claris, editors, *Aspects of Network and System Theory*, pages 385–407. New York: Holt, Rinehart and Winston, 1971.

[64] R. Kalman, P. Falb, and M. Arbib. *Topics in Mathematical System Theory*. McGraw-Hill, New York, 1969.

[65] W. Karush. *Minima of Functions of Several Variables with Inequalities as Side Constraints*. PhD thesis, University of Chicago, Chicago, 1939.

[66] G. Kotsalis, A. Megretski, and M. Dahleh. A model reduction algorithm for hidden Markov models. In *Proc. of the 46th IEEE Conference on Decision and Control*, pages 3424–3429, 2006.

[67] A. Krogh, M. Brown, I. Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994.

[68] A. Krogh, S. Mian, and D. Haussler. A hidden Markov model that finds genes in E. Coli DNA. *Nucleic Acids Research*, 22:4768–4778, 1994.

[69] H. Kuhn and A. Tucker. Nonlinear programming. In *Proc. of the 2nd Berkeley Symposium*, pages 481–492, 1951.

[70] W. Kuich and A. Salomaa. Semirings, automata, languages. In *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1986.

[71] S. Kung. A new identification and model reduction algorithm via singular value decomposition. In *Proc. of the 12th Asilomar Conference on Circuits, Systems and Computers*, pages 705–714, 1978.

[72] D. Lee and S. Sueng. Learning the parts of object by nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

[73] D. Lee and S. Sueng. Algorithms for nonnegative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562, 2001.

[74] T. Li and C. Ding. The relationships among various nonnegative matrix factorization methods for clustering. In *Proc. IEEE International Conference on Data Mining*, pages 362–371, 2006.

[75] L. Ljung. *System Identification, Theory for the user*. Prentice Hall, Upper Saddle River, New Jersey, 1999.

[76] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Wiley and Sons, New York, 1996.

[77] R. O'Flanagan, G. Paillard, R. Lavery, and A. Sengupta. Non-additivity in protein-DNA binding. *Bioinformatics*, 21:2254–2263, 2005.

[78] P. Van Overschee and B. De Moor. *Subspace Identification for Linear Systems: Theory, Implementation, Applications.* Kluwer Academic Publishers, 1996.

[79] P. Paatero and U. Tapper. Positive matrix factorization: A nonnegative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

[80] V. Pauca, J. Piper, and R. Plemmons. Nonnegative matrix factorization for spectral data analysis. *Linear Algebra and its Applications*, 416:29–47, 2006.

[81] A. Paz. *Introduction to Probabilistic Automata.* Academic Press, New York, 1971.

[82] M. Petrecksky. *Realization theory of hybrid systems.* PhD thesis, Vrije Universiteit Amsterdam, 2006.

[83] T. Petrie. Probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 10:97–115, 1969.

[84] G. Picci. On the internal structure of finite-state stochastic processes. In R.R. Mohler and A. Roberti, editors, *Lecture notes in Economics and Mathematical Systems*, volume 162, pages 288–304. Springer-Verlag, Berlin, 1978.

[85] G. Picci, J. van den Hof, and J. van Schuppen. Primes in several classes of the positive matrices. *Linear Algebra and its Applications*, 58:149–185, 1984.

[86] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 72:257–285, 1989.

[87] L. Rabiner and B. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3:4–16, 1986.

[88] J. Renegar. On the computational complexity and geometry of the first-order theory of reals, parts I, II, III. *Journal of Symbolic Logic*, 13:255–352, 1992.

[89] H. Rosenbrock. *State-space and Multivariable Theory.* Thomas Nelson and Sons, London, 1970.

[90] S. Salzberg, A. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26:544–548, 1998.

[91] A. Seidenberg. A new decision method for elementary algebra. *Annals of Mathematics*, 60:365–374, 1954.

[92] F. Shahnaz, M. Berry, V. Pauca, and R. Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing and Management*, 42:373–386, 2006.

[93] L. Silverman. Realization of linear dynamical systems. *IEEE Transactions on Automatic Control*, 16, 1971.

[94] B. Singh. Computational modeling and prediction of the human immunodeficiencyvirus (HIV) strains. In *Proc. of the IEEE International Joint Symposia on Intelligence and Systems*, pages 84–91, 1998.

[95] G. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35:551–566, 1993.

[96] M. Sznaier, O. Camps, and C. Mazzaro. Finite horizon model reduction of a class of neutrally stable systems with applications to texture synthesis and recognition. In *Proc. of the 43rd IEEE Conference on Decision and Control*, pages 3068–3073, 2004.

[97] A. Tarski. *A Decision Method for Elementary Algebra and Geometry, 2nd ed.* Univ. of California Press, Berkeley, 1951.

[98] A. Tether. Construction of minimal linear state-variable models from input-output data. *IEEE Transactions on Automatic Control*, 15:427–436, 1970.

[99] J. van den Hof. Realization of continuous-time positive systems. *Systems and Control Letters*, 31:243–253, 1997.

[100] J. van den Hof and J. van Schuppen. Positive matrix factorization via extremal polyhedral cones. *Linear Algebra and its Applications*, 293:171–186, 1999.

[101] A. Vandendorpe. *Model Reduction of Linear Systems, an Interpolation Point of View.* PhD thesis, Faculté des sciences appliquées, Université catholique de Louvain, Belgium, 2004.

[102] B. Vanluyten, K. De Cock, J. C. Willems, and B. De Moor. Equivalence of state representations for hidden Markov models. In *Proc. of the European Control Conference 2007 (ECC 2007)*, 2007.

[103] B. Vanluyten, J. C. Willems, and B. De Moor. Model reduction of systems with symmetries. In *Proc. of the 44th IEEE Conference on Decision and Control*, pages 826–831, 2005.

[104] B. Vanluyten, J. C. Willems, and B. De Moor. Matrix factorization and stochastic state representations. In *Proc. of the 45th IEEE Conference on Decision and Control*, pages 4188–4193, 2006.

[105] B. Vanluyten, J. C. Willems, and B. De Moor. A new approach for the identification of hidden Markov models. In *Proc. of the 46th IEEE Conference on Decision and Control*, 2007.

[106] B. Vanluyten, J. C. Willems, and B. De Moor. Approximate realization and estimation for quasi hidden Markov models. Submitted for publication, 2008.

[107] B. Vanluyten, J. C. Willems, and B. De Moor. Equivalence of state representations for hidden Markov models. *Systems and Control Letters*, 57:410–419, 2008.

[108] B. Vanluyten, J. C. Willems, and B. De Moor. Nonnegative matrix factorization without nonnegativity constraints on the factors. Submitted for publication, 2008.

[109] B. Vanluyten, J. C. Willems, and B. De Moor. Structured nonnegative matrix factorization with applications to hidden Markov realization and filtering. Accepted for publication in Linear Algebra and its Applications, 2008.

[110] B. Vanluyten, J.C. Willems, and B. De Moor. Recursive filtering using quasi-realizations. In C. Commault and N. Marchand, editors, *Lecture Notes in Control and Information Sciences*, volume 341, pages 367–374. Springer-Verlag, Berlin, 2006.

[111] S. Vasasis. On the complexity of nonnegative matrix factorization. Submitted for publication, 2007.

[112] M. Vidyasagar. The realization problem for hidden Markov models: The complete realization problem. In *Proc. of the 44th IEEE Conference on Decision and Control*, 2005.

[113] M. Vidyasagar. A realization theory for hidden Markov models: the complete realization problem. Submitted for publication, 2005.

[114] M. Vidyasagar. A realization theory for hidden Markov models: the partial realization problem. In *Proc. of the 17th International Symposium on Mathematical Theory of Networks and Systems*, pages 2145–2150, 2006.

[115] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[116] W.W. Wasserman and J.W. Fickett. Identification of regulatory regions which confer muscle-specific gene expression. *Journal of Molecular Biology*, 278:167–181, 1998.

[117] M. Welling and M. Weber. Positive tensor factorization. *Pattern Recognition Letters*, 22:1255–1261, 2001.

[118] H. Zeiger and A. McEwen. Approximate linear realizations of given dimension via ho's aalgorithm. *IEEE Transactions on Automatic Control*, 19:153, 1974.

[119] Q. Zhou and W.H.Wong. Cismodule: De novo discovery of cis-regulatory modules by hierarchical mixture modeling. *Proceedings of the National Academy of Sciences of the United States of Amerika*, 101:12114–12119, 2004.

Scientific curriculum vitae

Bart Vanluyten was born on Januari 18, 1980 in Leuven, Belgium. He went to the VIA College in Tienen and finished Latin-Mathematics in 1998. In July 2003, he received the degree of *Burgerlijk Werktuigkundig-Elektrotechnisch Ingenieur* (Electrical Engineer), option Data Mining and Automation, at the Katholieke Universiteit Leuven. His master thesis is entitled "A Dynamic Multi-Camera Solution for Tracking of Human Faces" and was carried out together with Stijn Wuyts. In October 2003, he started pursuing a Ph.D. on the subject of realization, identification and filtering for hidden Markov models in the SCD research group of the Department of Electrical Engineering (ESAT), under the supervision of Prof.dr.ir. Bart De Moor and Prof.dr.ir. Jan C. Willems. His research has been supported by the *Fonds voor Wetenschappelijk Onderzoek Vlaanderen.*

Publication list

Journal Papers

- B. Vanluyten, J.C. Willems and B. De Moor. Recursive Filtering using Quasi-Realizations. *Lecture Notes in Control and Information Sciences*, 341, 367–374, 2006.
- B. Vanluyten, J.C. Willems and B. De Moor. Equivalence of State Representations for Hidden Markov Models. *Systems and Control Letters*, 57(5), 410–419, 2008.
- B. Vanluyten, J.C. Willems and B. De Moor. Structured Nonnegative Matrix Factorization with Applications to Hidden Markov Realization and Filtering. *Accepted for publication in Linear Algebra and its Applications*, 2008.
- B. Vanluyten, J.C. Willems and B. De Moor. Nonnegative Matrix Factorization without Nonnegativity Constraints on the Factors. Submitted for publication.
- B. Vanluyten, J.C. Willems and B. De Moor. Approximate Realization and Estimation for Quasi hidden Markov models. Submitted for publication.

International Conference Papers

- I. Goethals, B. Vanluyten, B. De Moor Reliable spurious mode rejection using self learning algorithms. In *Proc. of the International Conference on Modal Analysis Noise and Vibration Engineering (ISMA 2004)*, Leuven, Belgium, pages 991–1003, 2004.
- B. Vanluyten, J. C. Willems and B. De Moor. Model Reduction of Systems with Symmetries. In *Proc. of the 44th IEEE Conference on Decision and Control (CDC 2005)*, Seville, Spain, pages 826–831, 2005.
- B. Vanluyten, J. C. Willems and B. De Moor. Matrix Factorization and Stochastic State Representations. In *Proc. of the 45th IEEE Conference*

on Decision and Control (CDC 2006), San Diego, California, pages 4188-4193, 2006.

- I. Markovsky, J. Boets, B. Vanluyten, K. De Cock, B. De Moor. When is a pole spurious? In *Proc. of the International Conference on Noise and Vibration Engineering (ISMA 2007)*, Leuven, Belgium, pp. 1615–1626, 2007.

- B. Vanluyten, J. C. Willems and B. De Moor. Equivalence of State Representations for Hidden Markov Models. In *Proc. of the European Control Conference 2007 (ECC 2007)*, Kos, Greece, 2007.

- B. Vanluyten, J. C. Willems and B. De Moor. A new Approach for the Identification of Hidden Markov Models. In *Proc. of the 46th IEEE Conference on Decision and Control (CDC 2006)*, New Orleans, Louisiana, 2007.

Internal Reports

- B. Vanluyten, K. De Cock, B. De Moor. Image Transformation and Compression using Realization Theory and Balanced Model Reduction. Internal Report 05-42, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2005.