KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee)

Bayesian learning with expert knowledge: Transforming informative priors between Bayesian networks and multilayer perceptrons

Promotoren:
Prof. dr. ir. Bart De Moor
Prof. dr. ir. Joos Vandewalle

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Geert FANNES

Juni 2004

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee)

Bayesian learning with expert knowledge: Transforming informative priors between Bayesian networks and multilayer perceptrons

Jury:
Prof. dr. ir. Ludo Froyen, voorzitter
Prof. dr. ir. Bart De Moor, promotor
Prof. dr. ir. Joos Vandewalle, promotor
Prof. dr. Jan Beirlant
Prof. dr. Désiré Bollé
Prof. dr. ir. Yves Moreau
Prof. dr. Dirk Timmerman
Prof. dr. ir. Sabine Van Huffel

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Geert FANNES

Juni 2004

© Katholieke Universiteit Leuven – Faculteit Toegepaste Wetenschappen
Arenbergkasteel, B-3001 Heverlee (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotocopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.

Voorwoord

Na het schrijven van mijn licentiaatsthesis, had ik haast gezworen om er nooit nog een te schrijven. Maar de interessante materie waar ik toen een glimp van mocht opvangen, kon me niet weerhouden. Er werd me immers de kans geboden om haast naadloos verder te werken op een onderwerp dat me altijd interesseerde, hetgeen ik me geen twee keer liet zeggen.

Ik kwam dan ook in een reeds vertrouwde omgeving terecht; ik kon blijven rekenen op Prof. Joos Vandewalle — mijn promotor van indertijd —, aangevuld met de inspirerende en motiverende kracht van Prof. Bart De Moor. Graag zou ik mijn beide promotoren willen bedanken voor de kansen die ze me geboden hebben en het vertrouwen dat ze in me stelden. Zonder enige aarzeling namen ze me aan, hoewel ik uit een volledig ander nest kwam en bitterweing wist over typische ingenieurszaken.

Ook mijn assessoren en leden van de jury zou ik willen bedanken voor de tijd en moeite die ze geïnvesteerd hebben in het lezen van dit werk en hun kritisch oordeel hierover. Ik het bijzonder wil ik Yves Moreau bedanken omdat hij de dagelijkse begeleiding op zich heeft genomen en zich zweet noch moeite heeft gespaard om me te helpen met het maken van deze tekst tot wat hij is. Indertijd heeft hij me uit de brand geholpen tijdens mijn licentiaatsthesis, en nu was zijn hulp eveneens van onschatbare waarde.

Further, the research that is described in this thesis would not have been possible without the valuable help of my friend and colleague Peter Antal, with who I worked on a daily basis. I will never forget the endless nights at ESAT debugging code and performing distributed computations manually on all the computers of Data4s at once. Peter, thanks a lot.

Ook Prof. Timmerman van het Departement Gynaecologie en Verloskunde van de Katholieke Universiteit Leuven ben ik veel dank verschuldigd. Niet alleen was hij bereid om zijn dataset met waardevolle gegevens uit handen te geven, hij verschafte mij en Peter ook het nodige inzicht in het ovariale tumor probleem en vulde ellenlange lijsten met vragen in over bepaalde kansen aangaande ovariale tumoren. Het was vooral een plezier om mee samen te werken.

Ook mijn collega's en vrienden verdienen een plaats in dit voorwoord. Op hen kon ik steeds rekenen zowel op wetenschappelijk als niet wetenschappelijk vlak. Ondanks het regelmatig gebrek aan tijd langs mijnentwege en de soms hectische toestanden, zijn zij diegenen die er altijd waren en met hen heb ik goede tijden beleefd.

Zij waaraan ik het meeste te danken heb, zijn ongetwijfeld mijn ouders. Doorheen de jaren hebben ze me steeds de beste kansen en mogelijkheden gegeven en zijn ze in mij blijven geloven. Ik vondt het dan ook niets minders dan mein pligt om hen een aantal uuren bezich te houden met het vindten van tallose spelingzvauten om mijn dank uit te drucken.

Tenslotte zou ik Annemie wel twintig keer willen bedanken. Zij was mijn grootste steun tijdens dit doctoraat-zonder-einde, terwijl ze haar geduld op wonderlijke wijze steeds wist te bewaren. Louter en alleen dankzij haar vond ik de tijd en rust om aan mijn onderzoek te werken, afgewisseld met de nodige ontspanning.

Abstract

The research we described in this thesis deals with learning probabilistic models based on heterogeneous information. We focused on classification systems and used the problem of pre-operational classification of ovarian tumours as a real-world application. Different types of information are available concerning this problem, such as statistical data, expert knowledge, and electronic text documents discussing the medical domain.

We will describe the *a priori* knowledge using a *donor* probabilistic model. Unfortunately, this model is usually not suitable to learn from data. We would like to perform this learning from data using an *acceptor* model. This model features good learning characteristics from data, but has often limited options to incorporate prior knowledge. We would like to combine the good properties of each model to reach an efficient learning behaviour based on data while still being able to incorporate the prior knowledge.

We developed a method to *transform* the information that is contained in the donor model to the acceptor model using *virtual data sets*. We present this method in the *Bayesian framework*, which is ideally suited to describe knowledge about a certain system and specifies how we have to update this knowledge when new information is observed.

To deal with the ovarian tumour classification problem, we chose a *Bayesian network* as donor and a *multilayer perceptron* as acceptor model. The Bayesian network enables us to describe the expert knowledge or incorporate information concerning the connection between variables that we can find by analyzing textual documents. On the down side, this model uses discrete variables and contains many parameters, which hinders the learning. The multilayer perceptron on the other hand contains less parameters, treats continuous variables in a natural way and shows a better learning behaviour based on data. This comes at the expense of the ability to incorporate prior knowledge fluently.

The results we describe in this thesis indicate that a successful transformation of information from a Bayesian network to a multilayer perceptron is possible. A considerable amount of the work consisted in implementing the necessary models and algorithms to perform and validate this transformation. These models and algorithms are described, together with some implementational considerations.

Samenvatting

Het onderzoek in deze thesis beschrijft hoe machines kunnen leren met behulp van probabilistische modellen op basis van heterogene informatie. We hebben ons geconcentreerd op het leren van classificatiesystemen, met als praktische toepassing het preoperationele classificeren van ovariale tumoren. Verschillende soorten informatie zijn voorhanden omtrent dit probleem waaronder statistische data, expertinformatie en relevante documenten uit elektronische gegevensbanken.

Deze laatste twee, ook wel *a priori* kennis genoemd, kunnen we beschrijven met behulp van een probabilistisch *donormodel*. Doorgaans zal zulk een model echter niet vlot leren van data. Leren van data zouden we daarom graag doen met behulp van een *acceptormodel*. Een dergelijk model leert wel goed van data maar kan moeilijk a priori kennis in rekening nemen. Graag zouden we de goede eigenschappen van elk model willen combineren, om te komen tot een efficiënt leergedrag op basis van de data terwijl we ook de a priori kennis mee in rekening nemen.

We ontwikkelden een methode om de informatie van het donormodel te *transformeren* naar deze van het acceptormodel met behulp van *virtuele datasets*. Deze methode situeert zich in het *Bayesiaanse denkkader* hetgeen bij uitstek geschikt is om de kennis omtrent een systeem te specificeren en aangeeft hoe we deze kennis moeten aanpassen als er nieuwe informatie wordt ingeworven.

Om het classificatieprobleem van ovariale tumoren aan te vatten, kozen we een *Bayesiaans netwerk* als donormodel, terwijl een *meerlaags perceptron* dienst deed als acceptormodel. Het Bayesiaanse netwerk laat ons toe om zowel de expertkennis te beschrijven als verbanden tussen variabelen te gebruiken die we kunnen vinden door de tekstdocumenten te analyseren. Dit model maakt echter gebruik van discrete variabelen en bevat veel parameters, wat het leren van de parameters op basis van data bemoeilijkt. Het meerlaags perceptron daarentegen bevat veel minder parameters, kan op een natuurlijke manier overweg met continue variabelen en leert beter van data. Dit is echter ten koste van de mogelijkheid om achtergrondinformatie vlot te verrekenen.

De resultaten in deze thesis tonen aan dat een succesvolle transformatie van informatie van een Bayesiaans netwerk naar een meerlaags perceptron mogelijk is. Een aanzienlijk deel van het werk bestond uit het implementeren van de benodigde modellen en technieken om deze transformatie uit te voeren en te valideren.

Nederlandse samenvatting

Bayesiaans leren op basis van expertkennis: transformatie van informatieve verdelingen tussen Bayesiaanse netwerken en meerlaagse perceptrons

Inleiding

Zodra de mens zich realiseerde dat hij kon leren, heeft dit leren op zich hem geïntrigeerd. Het is dan ook ons sterk ontwikkeld leer- en denkvermogen dat ons van de andere diersoorten onderscheidt.

Er werd in het verleden al veel tijd en moeite geïnvesteerd in onderzoek naar leren en denken en hoe dit gesimuleerd kon worden. Rond 1769 kon Wolfgang von Kempelen iedereen nog om de tuin leiden door een schakende machine te presenteren met het uitzicht van een houten Turk. Jammer genoeg berustte dit op puur boerenbedrog want er zat een dwerg verborgen in de machine. Toch duidt dit aan dat men toen een mechanische, schakende machine niet als volstrekt onmogelijk achtte.

Het is pas vanaf 1949, met de ontwikkeling van de computer door John von Neumann, dat de deur richting schaakspelende computers werd opengezet; tegenwoordig gelooft niemand nog dat een *volledig mechanische* schaakspelende machine gebouwd kan worden. De computer wordt algemeen wél aanzien als een beloftevol toestel om leergedrag te simuleren. De duidelijkste indicatie hiervan werd waarschijnlijk gegeven toen Deep Blue in 1997 Garry Kasparov, de toenmalige wereldkampioen schaken, versloeg.

Tegenwoordig noemt men "leren met behulp van de computer" *machine learning* of *artificiële intelligentie*. Het leerprobleem wordt vanuit verschillende uitgangspunten benaderd. In kansrekening wordt leren beschouwd als het proces om gekende informatie van een systeem in een model te omschrijven en te updaten. In functieapproximatie ligt de nadruk meer op het benaderen van een zekere meerdimensionale functie op basis van een aantal voorbeeldafbeeldingen. Andere methoden, zoals genetische algoritmen of neurale netwerken, proberen dan weer gekende biologische processen zoals de genetische hercombinatie (crossover en mutatie) of de cellulaire communicatie die optreedt in de hersenen te imiteren, en dit met een variërend gevoel voor realiteit.

De meeste van deze methoden zijn echter toegespitst op het verwerken van één specifiek type informatie, zoals numerieke representaties van karakterafbeeldingen voor geschriftsherkenning of geluidsbestanden voor spraakherkenning. Hoe we verschillende types informatie kunnen combineren in één model is nog steeds een vrij open probleem, en vormt het centrale thema van deze thesis.

Om het contact met de realiteit niet te verliezen, selecteerden we een medisch classificatieprobleem waarbij we trachten preoperatief de kwaadaardigheid van een ovariale tumor te voorspellen. In dit probleem onderscheiden we drie verschillende informatiebronnen die we wensen te combineren. Zo is er een statistische dataset met patiëntgegevens voorhanden, de kennis en ervaring van een arts en de relevante medische literatuur.

Dit medisch probleem wordt kort toegelicht in de volgende sectie, tezamen met de informatiebronnen die voor handen zijn. Verder wordt er een techniek geïntroduceerd die zowel de expertkennis als de data aan kan. Deze techniek is gebaseerd op twee verschillende modellen waarbij het kennisgebaseerde model (een Bayesiaans netwerk) verantwoordelijk is voor het omschrijven van de expertkennis. Het tweede model (een meerlaags perceptron, ook wel neuraal netwerk genoemd) is meer data georiënteerd en zal verantwoordelijk zijn voor het leren op basis van de numerieke dataset.

We hebben een techniek ontwikkeld waarbij de informatie van het Bayesiaanse netwerk *getransformeerd* naar het meerlaagse perceptron in de vorm van een *informatieve* a priori verdeling. Deze wordt dan op haar beurt getransformeerd wordt naar de a posteriori verdeling op basis van de dataset. De voorgestelde techniek is gebaseerd op *virtuele datasets* om de informatie over te dragen.

Classificatie van ovariale tumoren

Hieronder vindt u een korte toelichting van het classificatieprobleem van ovariale tumoren en een introductie van de informatie die hieromtrent voorhanden is.

Ovariale tumoren

De ovaria, ook wel eierstokken genoemd, zijn twee amandelvormig organen die zich aan weerszijde van de baarmoeder bevinden (zie Figuur 2.1). Deze kleine organen produceren de menselijke eicellen. Jammer genoeg zijn zij ook vrij vatbaar voor het ontwikkelen van gezwellen en tumoren. Deze ovariale tumoren worden onderverdeeld in drie grote categorieën die elk overeenstemmen met de drie types van cellen die we aantreffen in een ovarium: epitheliale tumoren, kiemceltumoren en stromale tumoren.

Hetgeen echter van groter belang is voor de patiënt, is het gedrag van de tumor. We onderscheiden ruwweg twee soorten: *goedaardige* en *kwaadaardige*. Figuur 2.3 toont zowel een goedaardige (links) als een kwaadaardige tumor (rechts). De tumoren uit de eerste categorie kunnen tamelijk groot worden en hierdoor pijnlijk zijn, maar tasten nooit het omliggend weefsel aan en brengen geen uitzaaiingen met zich mee. Men kan ze in veel gevallen doen krimpen door het toedienen van bepaalde hormonen of andere medicatie, of ze kunnen verwijderd worden met behulp van een doorsnee chirurgische ingreep.

Kwaadaardige tumoren, ook wel kankers genoemd, hebben wél de neiging om uit te zaaien eens ze groot genoeg zijn en zijn daardoor vaak levensbedreigend. Deze tumoren vragen om een drastische en ingrijpende behandeling door een gynaecologische oncoloog.

Het grote verschil tussen het gedrag van beide types tumoren en hun respectievelijke behandeling, vereist om *preoperatief* te bepalen of een tumor al dan niet kwaadaardig is. Het ontwikkelen van een classificatiesysteem dat de gynaecologische expert hierin adviseert, staat centraal in deze thesis.

Dit onderzoek plaatst zich in het kader van het Internationale Consortium voor Ovariale Tumoranalyse (IOTA), een groep van meerdere centra die meewerken aan het ontwikkelen van machine learning modellen voor de preoperatieve classificatie van ovariale tumoren (`https://www.iota-group.org/`). Dit project werd in 1998 opgestart door Prof. Dr. Dirk Timmerman, een arts van het departement gynaecologie en verloskunde aan het Universitaire Ziekenhuis te Leuven. Partners in dit project zijn het Departement Electrotechniek van de K.U.Leuven (ESAT/SCD) en verscheidene ziekenhuizen over de wereld.

Informatiebronnen

We hebben drie verschillende soorten informatie ter onzer beschikking om een classificatiesysteem te construeren.

Klinische data

De eerste en meest belangrijke informatiebron is een klinische dataset met patiëntgegevens. Op het moment van dit onderzoek bevat deze databank gegevens van 1 152 personen en 1 346 tumoren.[1] Elke tumor wordt omschreven met behulp van 68 parameters. Met behulp van input selectie procedures en de kennis van Prof. Timmerman werden de 35 meest relevante parameters geselecteerd waarop onze experimenten gebaseerd zullen zijn.

Hiertussen vinden we *Pathology*, de binaire variabele die aangeeft of de tumor goedaardig of kwaadaardig is. Verder wordt een tumor beschreven door variabelen die de vorm en de doorbloeding van de tumor aangeven, het geneeskundige verleden van de patiënt en de meting van het serum *CA125* in het bloed.

Deze dataset bevat zowel discrete als continue variabelen. De univariate statistieken van deze variabelen zijn weergegeven in de Tabellen 2.4, 2.5 en 2.6. Een meer diepgaande bespreking van de variabelen kan gevonden worden in Appendix A.

---

[1] Alhoewel dit niet de uiteindelijke IOTA dataset zal zijn, werd deze dataset met patiëntgegevens onderworpen aan een kwaliteitscontrole en werden inconsistente waarden gecorrigeerd door Andrea Valek.

Expertkennis

Aanvullend op deze numerieke informatie, konden we eveneens beroep doen op de kennis en ervaring van Prof. Timmerman, een toonaangevend expert op het gebied van ultrasonore technieken voor ovariale tumoren en de oprichter van het IOTA project. Het gros van de patiëntgegevens werd door hem verzameld.

Prof. Timmerman heeft zijn kennis over ovariale tumoren gespecificeerd met behulp van een Bayesiaans netwerk gebaseerd op 11 variabelen. Zowel de structuur van dit netwerk (zie Figuur 2.7) als de bijhorende parameters werden gegeven.

Verder kon hij ook de paarsgewijze verbanden tussen alle variabelen karakteriseren (zie Figuur 2.8). In Sectie 5.5.1 wordt aangegeven hoe deze informatie kan gebruikt worden om een verdeling over de ruimte van Bayesiaanse netwerkstructuren te definiëren.

Tekstdocumenten

Tot slot bezorgde Prof. Timmerman ons een verzameling karakteristieke kernwoorden per variabele, een tekstuele omschrijving van deze variabelen en een selectie van relevante medische literatuur. Met behulp van deze aanwijzingen waren we in staat om ook de tekstuele informatie mee in rekening te nemen, eveneens in de vorm van een verdeling over de ruimte van Bayesiaanse netwerkstructuren.

Het Bayesiaanse denkkader

We wensen de heterogene informatie die in de vorige sectie geïntroduceerd werd, te combineren om tot een zo goed mogelijk classificatiemodel te komen. Om dit te verwezelijken, behandelen we het leerprobleem in het Bayesiaanse denkkader. Dit denkkader is uitermate geschikt om kennis over een bepaald systeem te specificeren en om deze kennis aan te passen als er nieuwe observaties binnenkomen.

De regel van Bayes

In het Bayesiaanse denkkader duidt een kans de *gradatie van geloof* aan over de waarheid van een bepaald statement. Dit geloof is *altijd* afhankelijk van de informatie waarover men kan beschikken en deze interpretatie kan zonder problemen toegepast worden op bepaalde dingen die niet intrinsiek random zijn; sommige dingen kunnen met zekerheid voorspeld worden, maar wanneer de benodigde informatie ontbreekt of de berekening te ingewikkeld is, moeten we onze toevlucht nemen tot plausibele redeneringen. Zo kunnen we bij een vogelpikspel perfect berekenen onder of iemand de roos zal raken, maar enkel als de initiële snelheids- en plaatsvectoren gekend zijn.

Richard Cox [20] toonde aan dat elk systeem dat gebruikt kan worden om consistent te leren en te redeneren onder een aantal basisvoorwaarden, steeds getransformeerd kan worden naar het gekende systeem van kansrekenen met als basisregels de som- en productregel:

$$\sum_a \mathrm{P}(\, \underline{a} = a \,|\, \xi \,) = 1 \quad \text{met} \quad \mathrm{P}(\, \underline{a} = a \,|\, \xi \,) \geq 0$$
$$\mathrm{P}(\, \underline{a} = a, \underline{b} = b \,|\, \xi \,) = \mathrm{P}(\, \underline{a} = a \,|\, \underline{b} = b, \xi \,)\, \mathrm{P}(\, \underline{b} = b \,|\, \xi \,).$$

De eerste basisvoorwaarde zegt dat de kans dat een bepaalde statement waar is, ons onmiddellijk leert wat de kans is dat dit statement niet waar is. De tweede voorwaarde legt op dat door het specificeren van de kans dat een bepaald statement waar is samen met het specificeren van de kans dat een tweede statement waar is gegeven dat ons eerste statement correct is, we de kans kennen dat beide statements correct zijn. Tenslotte bedoelen we met consistent leren dat het gebruik van dezelfde informatie op verschillende wijzen tot eenzelfde resultaat moet leiden. Met het symbool ξ noteren we alle achtergrondinformatie die voorhanden is.

Een van de meest gebruikte regels binnen de kansrekening, is de *regel van Bayes*:

$$\mathrm{P}(\, \underline{a} = a \,|\, \underline{b} = b \,) = \frac{\mathrm{P}(\, \underline{b} = b \,|\, \underline{a} = a \,)\, \mathrm{P}(\underline{a} = a)}{\mathrm{P}(\underline{b} = b)} \tag{1}$$

$$\begin{aligned} \mathrm{p}(\, \boldsymbol{\omega} \,|\, \mathbf{D}, \xi \,) &= \frac{\mathrm{p}(\, \mathbf{D} \,|\, \boldsymbol{\omega}, \xi \,)\, \mathrm{p}(\, \boldsymbol{\omega} \,|\, \xi \,)}{\mathrm{p}(\, \mathbf{D} \,|\, \xi \,)} \qquad (2) \\ &\propto \mathrm{p}(\, \mathbf{D} \,|\, \boldsymbol{\omega} \,)\, \mathrm{p}(\boldsymbol{\omega}). \end{aligned}$$

Hierboven is de regel van Bayes twee keer vermeld. Vergelijking 1 is de abstracte vorm en zegt niet veel over het gebruik van deze regel. Vergelijking 2 daarentegen duidt met symbolen aan waar deze regel veelal zijn toepassing vindt. Hier stelt $\boldsymbol{\omega}$ de parametervector van een bepaalde verdeling $\mathrm{p}(\, \cdot \,|\, \boldsymbol{\omega} \,)$ voor, $\mathbf{D}$ duidt de dataset aan en ξ bevat de achtergrondinformatie die voorhanden is.

Meestal zijn we geïnteresseerd hoe ons geloof in de verschillende modelparameters $\boldsymbol{\omega}$ (voorgesteld door de verdeling $\mathrm{p}(\, \boldsymbol{\omega} \,|\, \xi \,)$) beïnvloed wordt door het observeren van een dataset $\mathbf{D}$. De regel van Bayes leert ons dat de *a posteriori* verdeling $\mathrm{p}(\, \boldsymbol{\omega} \,|\, \mathbf{D}, \xi \,)$ (ná het observeren van de data) berekend kan worden door de data likelihood $\mathcal{L}(\, \boldsymbol{\omega} \,|\, \mathbf{D} \,) = \mathrm{p}(\, \mathbf{D} \,|\, \boldsymbol{\omega} \,)$ van de modelparameters te vermenigvuldigen met de *a priori* verdeling $\mathrm{p}(\, \boldsymbol{\omega} \,|\, \xi \,)$ (onze kennis over de parameters vóór het observeren van de data). De noemer $\mathrm{p}(\, \mathbf{D} \,|\, \xi \,)$ is onafhankelijk van de modelparameters $\boldsymbol{\omega}$ en wordt daarom vaak weggelaten.

De geschiedenis van het Bayesiaanse denkkader

De interpretatie van kansrekenen en haar toepasbaarheid was stof tot veel discussie in het verleden. Jacob Bernoulli (1654–1705) was een van de eerste die zich afvroeg hoe *inductief* redeneren verwezelijkt kon worden met behulp van *deductief* redeneren. Deze deductieve redeneerprocessen hebben tot doel om, vertrekkende van een bepaald begingegeven, verschillende mogelijk uitkomsten af te leiden. De meest voor de hand liggende voorbeelden vinden we in de exacte wiskunde waar men, uitgaande van een aantal axioma's, een bepaalde stelling probeert te bewijzen. De meeste kansspelen zijn een andere groep voorbeelden. Hier probeert men om, vertrekkende van een aantal goed gedefinieerde — maar moeilijk te vinden — voorwerpen zoals eerlijke dobbelstenen, kansen toe te kennen aan bepaalde observaties.

Het inductieve redeneren beoogt het tegenovergestelde van deductief redeneren en probeert op basis van een aantal observaties of uitkomsten de beginoorzaken te achterhalen, zoals het wel of niet eerlijk zijn van een dobbelsteen. Elke dag worden we met zulke vraagstukken geconfronteerd. Hoewel de mens hier intuïtief en gemakkelijk mee omgaat, zijn deze problemen minder eenvoudig om op te lossen met behulp van een computer.

Thomas Bayes (1702–1762) [8] vond het antwoord op Bernoulli's probleem en zijn resultaten werden verder uitgewerkt en toegepast door Pierre-Simon Laplace (1749–1827) [57]. Volgens Bayes en Laplace stelde een kans een bepaalde *gradatie van geloof* of *plausibiliteit* voor dat een bepaald statement waar is.

Voor veel wetenschappers was dit filosofische concept van kans veel te vaag en te subjectief. John Venn (1834–1923) stelde daarom een nieuwe definitie voor op basis van de *relatieve frequentie* van een bepaalde gebeurtenis bij veelvuldige herhaling, in een poging om de definitie objectiever te maken. Dit wordt ook wel de frequentistische kijk op kansrekenen genoemd. Dit concept kan echter moeilijk toegepast worden op dingen die intrinsiek niet random zijn, zoals bijvoorbeeld de massa van een planeet. Om het toch mogelijk te maken iets te kunnen zeggen over de massa van een planeet op basis van astronomische meetgegevens wordt de massa van de planeet gerelateerd aan de meetgegevens met behulp van een *statistiek*. De massa is een constante, maar op de data zit wel meetruis, waardoor de waarde van de statistiek wél een random variabele wordt waarop men de frequentistische kansrekening van John Venn kan toepassen. Eén van de belangrijkste mensen op het vlak van de statistiek was ongetwijfeld sir Ronald Fisher (1890–1962). Hij ontwikkelde onder meer de maximum likelihoodtechniek, de analyse van de variantie (ANOVA) en concepten als sufficiëntie.

Door het werk van sir Harold Jeffreys (1891–1989) [48] is het nu weer natuurlijker om een kans te interpreteren als het gebrek aan kennis over een bepaalde systeem om een uitspraak met zekerheid te doen. Dit "gebrek aan kennis" concept is equivalent aan de plausibiliteitsdefinitie van Bayes en Laplace. Richard Cox (1898–1991) [20] kon tenslotte aantonen dat, vertrekkende van de plausibiliteitsdefinitie, kans theorie het enige systeem is dat hiervoor gebruikt kan worden. Deze resultaten werden recentelijk verder uitgewerkt door Edwin Jaynes (1922–1998) [47].

De a priori verdeling

De a priori verdeling $\mathrm{p}(\boldsymbol{\omega} \mid \xi)$ heeft tot doel onze kennis over de parameters van een bepaald model $\mathrm{p}(\cdot \mid \boldsymbol{\omega})$ weer te geven. Vaak wordt deze a priori verdeling uit een speciale familie van verdelingen gekozen met het oog op het berekenen van de a posteriori verdeling. Uit Vergelijking 2 blijkt immers dat deze a posteriori verdeling in essentie het product is van de a priori verdeling en de data likelihood.

Indien dit product resulteert in een verdeling van dezelfde familie als de a priori verdeling spreken we van een verdeling die *toegevoegd* is aan de data verdeling. Eén van de meest frequente voorbeelden is de Dirichlet verdeling, welke toegevoegd is aan de tabelverdeling (zie Sectie 5.3.1). De keuze van deze verdeling is niet enkel gemotiveerd vanuit praktische overwegingen; Heckerman [41] toont aan dat dit de enige redelijke keuze.

Eens we de familie voor onze a priori verdeling gekozen hebben, moeten we nog een bepaalde verdeling uit deze familie kiezen door de *hyper*parameters te specificeren. Deze hyperparameters zijn de parameters van de a priori verdeling. Hier onderscheiden we twee mogelijke keuzes: als eerste kunnen we een *regulerende* a priori verdeling kiezen. Deze verdelingen worden voornamelijk gebruikt wanneer een parametervector $\boldsymbol{\omega}$ met een grote norm een meer complex model voorstelt. Als onze a priori verdeling de norm van de parametervector restricteert, bekomen we een regulerend effect. Deze verdelingen bevatten doorgaans geen specifieke informatie over het probleem dat we wensen te modelleren en presteren daardoor even slecht als een random model wanneer nog geen data geobserveerd is. Deze verdelingen worden *niet-informatieve* complexiteitsgebaseerde verdelingen genoemd.

Een tweede type verdelingen bevat *wel* specifieke informatie over het te modelleren probleem en worden daardoor *informatieve* verdelingen genoemd.

Classificatie

Eens we de a priori verdeling gespecificeerd hebben geeft de regel van Bayes ons de a posteriori verdeling. Deze a posteriori verdeling drukt ons vertrouwen uit in de verschillende mogelijke parametrisaties $\boldsymbol{\omega}$, nadat we zowel de achtergrondkennis ξ als de dataset $\mathbf{D}$ kennen. Toegepast op een binair classificatieprobleem, bekomen we de volgende formule:

$$\mathrm{P}(\,\underline{t} = \mathcal{C}_{\mathrm{P}} \,|\, \boldsymbol{x}, \mathbf{D}, \xi\,) = \int_{\boldsymbol{\Omega}} \mathrm{P}(\,\underline{t} = \mathcal{C}_{\mathrm{P}} \,|\, \boldsymbol{x}, \boldsymbol{\omega}\,)\, \mathrm{p}(\,\boldsymbol{\omega} \,|\, \mathbf{D}, \xi\,)\, d\boldsymbol{\omega}.$$

Hierbij is $\underline{t}$ het binaire classificatielabel van een record met observaties $\boldsymbol{x}$ dat we wensen te classificeren als $\mathcal{C}_{\mathrm{N}}$ of $\mathcal{C}_{\mathrm{P}}$ (een negatieve of positieve classificatie). Verder stelt $\mathbf{D}$ een dataset voor met gekende klasselabels.

Met behulp van bovenstaande kans en een classificatiedrempelwaarde λ kunnen we een beslissing nemen:

$$\underline{t} = \begin{cases} \mathcal{C}_{\mathrm{P}} & \text{als } \mathrm{P}(\,\underline{t} = \mathcal{C}_{\mathrm{P}} \,|\, \boldsymbol{x}, \mathbf{D}, \xi\,) \geq \lambda, \\ \mathcal{C}_{\mathrm{N}} & \text{anders.} \end{cases}$$

De ROC performantiemaat

In deze thesis gaan we verschillende classificatiesystemen ontwikkelen voor het tumorprobleem. Om deze verschillende modellen met elkaar te kunnen vergelijken, hebben we een performantie maat nodig. Alle resultaten die we in deze thesis presenteren, zijn gebaseerd op de *receiver operating characteristics* curve (ROC) [38]. Deze curve wordt aangemaakt door de *sensitiviteit* uit te zetten in functie van *1-specificiteit* voor een variërende classificatiedrempelwaarde en bevindt zich in het eenheidsvierkant $[0,1] \times [0,1]$. Figuur 3.8 toont twee voorbeelden van ROC curven.

De *oppervlakte* onder deze curve is een veelgebruikte performantiemaat die onafhankelijk is van de classificatiedrempelwaarde λ. Verder is deze oppervlakte gerelateerd aan de Wilcoxon statistiek en kan geïnterpreteerd worden als de kans dat $\mathrm{P}(\, \underline{t} = \mathcal{C}_\mathrm{P} \,|\, \boldsymbol{x}_\mathrm{N} \,)$ kleiner is dan $\mathrm{P}(\, \underline{t} = \mathcal{C}_\mathrm{P} \,|\, \boldsymbol{x}_\mathrm{P} \,)$, waarbij $\boldsymbol{x}_\mathrm{N}$ een willekeurige negatieve en $\boldsymbol{x}_\mathrm{P}$ een willekeurige positieve observatie is:

$$\mathrm{AUC} = \mathrm{P}(\, \mathrm{P}(\, \underline{t} = \mathcal{C}_\mathrm{P} \,|\, \boldsymbol{x}_\mathrm{N} \,) < \mathrm{P}(\, \underline{t} = \mathcal{C}_\mathrm{P} \,|\, \boldsymbol{x}_\mathrm{P} \,) \,|\, \boldsymbol{x}_\mathrm{N} \in \mathcal{C}_\mathrm{N} \text{ en } \boldsymbol{x}_\mathrm{P} \in \mathcal{C}_\mathrm{P} \,).$$

Transformatie van informatie representatie

Soms kunnen we niet alle informatie in"en model combineren. Het onderzoek in deze thesis laat toe om de a priori kennis te beschrijven met een geschikt model. Vervolgens wordt deze informatie getransformeerd naar een ander model, geschikt om verder te leren op basis van de data.

Donor- en acceptormodel

Het eerste model dat we nodig hebben, het *donormodel*, beschrijft een gezamelijke kansverdeling $\mathrm{p}(\, \cdot \,|\, \theta \,)$. We specificeren dit model met behulp van de achtergrondinformatie door de verdeling $\mathrm{p}(\, \theta \,|\, \xi \,)$ over de modelparameters te definiëren. We gaan ervan uit dat we de achtergrondinformatie vlot in rekening kunnen nemen. Meestal heeft deze eis tot gevolg dat de leercapaciteiten van deze modelklasse op basis van de data is eerder beperkt.

Daarom wensen we deze informatie te transformeren naar een *acceptormodelklasse* $\mathrm{p}(\, \cdot \,|\, \omega \,)$, welke wél goede leercapaciteiten op basis van data bezit. Het acceptormodel heeft dan weer beperkte mogelijkheden om rechtstreek a priori kennis in rekening te nemen.

Transformatie tussen donor- en acceptormodel

We proberen de positieve kanten van beide modellen te combineren door middel van *virtuele* datasets: we zijn geïnteresseerd in de a priori verdeling voor de parameters ω van het acceptormodel, gegeven de achtergrondinformatie:

$$\begin{aligned}
\mathrm{p}(\,\omega\,|\,\xi\,) &= \sum_{\mathbf{D}_k} \mathrm{p}(\,\omega\,|\,\mathbf{D}_k,\xi\,)\,\mathrm{p}(\,\mathbf{D}_k\,|\,\xi\,) \\
&= \sum_{\mathbf{D}_k} \mathrm{p}(\,\omega\,|\,\mathbf{D}_k,\xi\,) \int_\theta \mathrm{p}(\,\mathbf{D}_k\,|\,\theta,\xi\,)\,\mathrm{p}(\,\theta\,|\,\xi\,)\,d\theta \\
&\approx \sum_{\mathbf{D}_k} \mathrm{p}(\,\omega\,|\,\mathbf{D}_k,\xi_{\mathrm{c}}\,) \int_\theta \mathrm{p}(\,\mathbf{D}_k\,|\,\theta,\xi\,)\,\mathrm{p}(\,\theta\,|\,\xi\,)\,d\theta. \qquad (3)
\end{aligned}$$

We stellen met het symbool θ de parametrisatie van het donormodel voor, terwijl ω de parametrisatie van het acceptormodel is.

We sommeren over alle mogelijke datasets met k records. Deze datasets zijn afkomstig van de gezamelijke verdeling van het donormodel. In Vergelijking 3 gaan we ervan uit dat we de achtergrondinformatie ξ mogen vervangen door het complexiteitsgebaseerde gedeelte ξ_{c}, eens de dataset $\mathbf{D}_k$ gekend is.

Vergelijking 3 kunnen we rechtstreeks vertalen naar een algoritme om parametervectoren volgens $\mathrm{p}(\,\omega\,|\,\xi\,)$ te genereren:

1. Genereer een donorparametrisatie θ volgens de informatieve a priori verdeling $\mathrm{p}(\,\theta\,|\,\xi\,)$.
2. Genereer een virtuele dataset $\mathbf{D}_k$ volgens de gezamelijke verdeling gedefinieerd door het donormodel $\mathrm{p}(\,\cdot\,|\,\theta\,)$ met als parameters θ.
3. Genereer een parametervector ω volgens de a posteriori verdeling gebaseerd op de virtuele dataset $\mathrm{p}(\,\omega\,|\,\mathbf{D}_k,\xi_{\mathrm{c}}\,)$.

Het kiezen van het aantal records in elke dataset vergt enige voorzichtigheid. Deze parameter moet groot genoeg zijn zodat we geen al te grote fout maken door ξ te vervangen door ξ_{c}. Hoe groter we k kiezen, hoe accurater de transformatie zal verlopen. Toch mogen we k ook niet te groot nemen. Immers, hoe groter k wordt, hoe gepiekter de verdelingen $\mathrm{p}(\,\omega\,|\,\mathbf{D}_k\,)$ er zullen uitzien. Als nu geen enkele van de virtuele datasets $\mathbf{D}_k$ eruit ziet als de echte dataset $\mathbf{D}$ omdat ons donormodel bijvoorbeeld niet alle karakteristieken van $\mathbf{D}$ kan omschrijven, zal *geen enkele* a posteriori verdeling $\mathrm{p}(\,\omega\,|\,\mathbf{D}_k\,)$ de optimale acceptorparametrisatie in zijn drager hebben. Dit zou betekenen dat de bovenstaande informatieve a priori verdeling $\mathrm{p}(\,\omega\,|\,\xi\,)$ eveneens de optimale parametrisatie niet zal bevatten, hetgeen uiteraard een ongewenst gedrag is.

Bayesiaanse netwerken

De vorige sectie maakte gebruik van twee verschillende modelklassen en definieerde een overgang tussen beide. Deze sectie introduceert het donormodel, hetgeen we wensen te gebruiken om de achtergrondinformatie voor te stellen. Hiervoor opteerden we voor het *Bayesiaans netwerk*, een model dat de laatste jaren nogal wat aan populariteit heeft ingewonnen. Dit model laat toe om op een gefundeerde wijze achtergrondinformatie in rekening te nemen.

Structuur en parameters

In essentie is een Bayesiaans netwerk een methode om een *gezamelijke* verdeling op een spaarse wijze neer te schrijven. Deze is gebaseerd op de gekende kettingregel uit de kansrekening:

$$\begin{aligned} \mathrm{p}(\underline{x}_1, \ldots, \underline{x}_v) &= \mathrm{p}(\underline{x}_1)\, \mathrm{p}(\, \underline{x}_2 \,|\, \underline{x}_1 \,) \cdots \mathrm{p}(\, \underline{x}_v \,|\, \underline{x}_1, \ldots, \underline{x}_{v-1} \,) \\ &= \prod_{i=1}^{v} \mathrm{p}(\, \underline{x}_i \,|\, \underline{x}_1, \ldots, \underline{x}_{i-1} \,). \end{aligned} \tag{4}$$

Hierbij zijn $\underline{x}_1, \ldots, \underline{x}_v$ stochastische veranderlijken en is de factorisatie *afhankelijk* van de volgorde van de veranderlijken. Vertrekkende van Vergelijking 4 vereenvoudigen we elke factor afzonderlijk, afhankelijk van de verdeling die we modelleren. Zo kan het bijvoorbeeld zijn dat $\underline{x}_3$ conditioneel onafhankelijk is van $\underline{x}_1$ gegeven $\underline{x}_2$. Hiermee bedoelen we dat de extra informatie in $\underline{x}_1$ ons niets nieuws leert over $\underline{x}_3$ als $\underline{x}_2$ al gekend is. We noteren deze eigenschap door $(\, \underline{x}_3 \perp \underline{x}_1 \,|\, \underline{x}_2 \,)$. Dit laat ons toe om de derde factor te vervangen door een eenvoudigere factor

$$\mathrm{p}(\, \underline{x}_3 \,|\, \underline{x}_1, \underline{x}_2 \,) = \mathrm{p}(\, \underline{x}_3 \,|\, \underline{x}_2 \,).$$

Als we deze vereenvoudigingen voor elke factor bepalen, bekomen we

$$\begin{aligned} \mathrm{p}(\underline{x}_1, \ldots, \underline{x}_v) &= \prod_{i=1}^{v} \mathrm{p}(\, \underline{x}_i \,|\, \pi(\underline{x}_i) \,) \\ \pi(\underline{x}_i) &\subset \{\underline{x}_1, \ldots, \underline{x}_{i-1}\}, \end{aligned} \tag{5}$$

waarbij we de verzameling $\pi(\underline{x}_i)$ de *ouders* van de variabele $\underline{x}_i$ noemen.

Uit bovenstaande formulering kunnen we rechtstreeks de tweeledige natuur van een Bayesiaans netwerk afleiden: de conditionele onafhankelijkheden geven ons informatie over *welke* variabelen afhankelijk zijn van welke andere. *Hoe* deze afhankelijkheden er in de praktijk uitzien, wordt bepaald door de lokale afhankelijkheidsmodellen $\mathrm{p}(\, \underline{x}_i \,|\, \pi(\underline{x}_i) \,)$.

Het eerste type informatie kan voorgesteld worden met behulp van een *gerichte, niet-cyclische graf.* Hierdoor worden Bayesiaanse netwerken tot de klasse van *grafische* modellen gerekend. Deze netwerkvoorstelling wordt ook wel de *structuur* van het Bayesiaans netwerkmodel genoemd. Doorgaans komen we tot deze voorstelling door elke variabele met een knoop voor te stellen. De ouders worden verbonden met het kind door middel van een pijl die naar het kind wijst. Zo toont Figuur 5.1 de structuur van een model met vijf variabelen. Voor het linkse model waren geen vereenvoudigingen mogelijk terwijl het rechtse model correspondeert met de gezamelijke verdeling

$$\begin{aligned} \mathrm{p}(\underline{a}, \underline{b}, \underline{c}, \underline{d}, \underline{e}) &= \mathrm{p}(\underline{a})\, \mathrm{p}(\, \underline{b} \,|\, \underline{a} \,)\, \mathrm{p}(\, \underline{c} \,|\, \underline{a}, \underline{b} \,)\, \mathrm{p}(\, \underline{d} \,|\, \underline{a}, \underline{b}, \underline{c} \,)\, \mathrm{p}(\, \underline{e} \,|\, \underline{a}, \underline{b}, \underline{c}, \underline{d} \,) \\ &= \mathrm{p}(\underline{a})\, \mathrm{p}(\underline{b})\, \mathrm{p}(\underline{c})\, \mathrm{p}(\, \underline{d} \,|\, \underline{a}, \underline{b} \,)\, \mathrm{p}(\, \underline{e} \,|\, \underline{c}, \underline{d} \,). \end{aligned}$$

Het tweede type informatie, de lokale afhankelijkheidsmodellen, worden ook wel de *parameters* van het netwerk genoemd en bepalen hoe de effectieve afhankelijkheden eruit zien.

Van zodra we de structuur van ons model bepaald hebben, samen met de parameters van de lokale afhankelijkheidsmodellen, hebben we de gezamelijke kansverdeling volledig gespecificeerd. De volgende stap bestaat er dan meestal in om bepaalde *vragen* te stellen aan deze verdeling. Deze vragen kunnen meestal geformuleerd worden in de vorm van een bepaalde marginale of conditionele verdeling. Zo zijn we in een classificatiesetup meestal geïnteresseerd in de kansverdeling van de klassevariabele $\underline{t}$, gegeven een aantal metingen of symptomen (niet noodzakelijk allemaal)

$$\mathrm{p}(\,\underline{t}\,|\,\underline{a}=a,\underline{c}=c\,).$$

Vooraleer we hier aan toe zijn, moeten we de structuur en de parameters van ons model bepalen.

Het leren van de structuur

De structuur kan men manueel specificeren of aanleren met behulp van een dataset. Deze laatste methode zoekt naar een netwerkstructuur $\mathcal{S}_{\mathrm{bn}}$ met een hoge a posteriori kans $\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}}\,|\,\mathbf{D},\xi\,)$. Als we ervan uit gaan dat we a priori geen voorkeur hebben voor een bepaalde structuur, dan kunnen we $\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}}\,|\,\mathbf{D},\xi\,)$ op volgende wijze uitwerken:

$$\begin{aligned}
\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}}\,|\,\mathbf{D},\xi\,) &= \frac{\mathrm{p}(\,\mathbf{D}\,|\,\mathcal{S}_{\mathrm{bn}}\,)\,\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}}\,|\,\xi\,)}{\mathrm{p}(\mathbf{D})} \\
&\propto \mathrm{p}(\,\mathbf{D}\,|\,\mathcal{S}_{\mathrm{bn}}\,) \\
&= \int_{\boldsymbol{\Theta}} \mathrm{p}(\,\mathbf{D}\,|\,\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}\,)\,\mathrm{p}(\boldsymbol{\theta})\,d\boldsymbol{\theta} \\
&= \int_{\boldsymbol{\Theta}} \prod_{i=1}^{n} \mathrm{p}(\,\boldsymbol{x}^i\,|\,\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}\,)\,\mathrm{p}(\boldsymbol{\theta})\,d\boldsymbol{\theta} \\
&= \int_{\boldsymbol{\Theta}} \prod_{i=1}^{n}\prod_{j=1}^{v} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i),\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}\,)\,\mathrm{p}(\boldsymbol{\theta})\,d\boldsymbol{\theta} \\
&= \int_{\boldsymbol{\Theta}} \prod_{j=1}^{v}\prod_{i=1}^{n} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i),\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}_j\,)\,\mathrm{p}(\boldsymbol{\theta})\,d\boldsymbol{\theta}.
\end{aligned}$$

We hebben de constante factoren $\mathrm{p}(\mathbf{D})$ en $\mathrm{p}(\mathcal{S}_{\mathrm{bn}})$ weggelaten en gaan ervan uit dat we de parameters $\boldsymbol{\theta}$ kunnen verdelen over de lokale afhankelijkheidsmodellen;

$$\mathrm{p}(\,\underline{x}_j\,|\,\pi(\underline{x}_j),\boldsymbol{\theta}\,) = \mathrm{p}(\,\underline{x}_j\,|\,\pi(\underline{x}_j),\boldsymbol{\theta}_j\,).$$

Als we verder nog aannemen dat deze verschillende parametergroepen a priori onafhankelijk zijn

$$\mathrm{p}(\boldsymbol{\theta}) = \prod_{j=1}^{v} \mathrm{p}(\boldsymbol{\theta}_j),$$

dan kunnen we onze afleiding verder zetten:

$$\begin{aligned}
\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}} \,|\, \mathbf{D}\,) &\propto \int_{\mathbf{\Theta}} \prod_{j=1}^{v} \prod_{i=1}^{n} \mathrm{p}(\, x_j^i \,|\, \pi(x_j^i), \mathcal{S}_{\mathrm{bn}}, \boldsymbol{\theta}_j \,) \prod_{k=1}^{v} \mathrm{p}(\boldsymbol{\theta}_k)\, d\boldsymbol{\theta} \\
&= \int_{\mathbf{\Theta}} \prod_{j=1}^{v} \prod_{i=1}^{n} \mathrm{p}(\, x_j^i \,|\, \pi(x_j^i), \mathcal{S}_{\mathrm{bn}}, \boldsymbol{\theta}_j \,)\, \mathrm{p}(\boldsymbol{\theta}_j)\, d\boldsymbol{\theta} \\
&= \prod_{j=1}^{v} \int_{\mathbf{\Theta}_j} \prod_{i=1}^{n} \mathrm{p}(\, x_j^i \,|\, \pi(x_j^i), \mathcal{S}_{\mathrm{bn}}, \boldsymbol{\theta}_j \,)\, \mathrm{p}(\boldsymbol{\theta}_j)\, d\boldsymbol{\theta}_j \\
&= \prod_{j=1}^{v} L(\, j \,|\, \pi_j\,).
\end{aligned}$$

Deze laatste vergelijking duidt aan dat de verdeling over de netwerkstructuren ontbonden kan worden, waarbij elke factor $L(\, j \,|\, \pi_j\,)$ de kans van een lokale substructuur $\underline{x}_j$ met ouders π_j voorstelt. Dit laat ons toe om het *maximum a posteriori* netwerk te zoeken door voor elke variabele *afzonderlijk* de optimale ouders te zoeken. Deze lokale zoektocht naar een goede set van ouders kan op een exhaustatieve manier gebeuren of met behulp van een *gulzige* heuristiek. Deze laatste methode test niet alle mogelijke oudercombinaties uit, maar zoekt een ouderverzameling door telkens díe knoop toe te voegen die de kans van de lokale substructuur het meeste doet toenemen.

Deze structuurleermethoden vertrekken van een *vaste* volgorde van de variabelen waarop de kettingregel van de kansrekening wordt toegepast. Indien deze volgorde niet gekend is, wat vaak voorkomt, wordt het bovenstaande structuurleeralgoritme verschillende malen toegepast op willekeurige permutaties van de variabelen. We hebben dit uitgebreid naar een techniek om een structuur te leren in combinatie met een volgorde voor de variabelen, zoals geïntroduceerd op het einde van in Sectie 5.2.1.

Het leren van de parameters

Eens de structuur van ons Bayesiaanse netwerk gekend is, kunnen we overgaan tot het leren van de parameters. We zullen deze parameters behandelen in het Bayesiaanse denkkader, waar we de informatie die over deze parameters gekend is, voorstellen met behulp van een verdeling. Deze verdeling kan eveneens gefactoriseerd worden per lokale substructuur:

$$\begin{aligned}
\mathrm{p}(\,\boldsymbol{\theta} \,|\, \mathcal{S}_{\mathrm{bn}}, \mathbf{D}\,) &= \frac{\mathrm{p}(\,\mathbf{D} \,|\, \boldsymbol{\theta}, \mathcal{S}_{\mathrm{bn}}\,)\, \mathrm{p}(\,\boldsymbol{\theta} \,|\, \mathcal{S}_{\mathrm{bn}}\,)}{\mathrm{p}(\,\mathbf{D} \,|\, \mathcal{S}_{\mathrm{bn}}\,)} \\
&\propto \prod_{j=1}^{v} \left(\prod_{i=1}^{n} \mathrm{p}(\, x_j^i \,|\, \pi(x_j^i), \boldsymbol{\theta}_j, \mathcal{S}_{\mathrm{bn}}\,)\, \mathrm{p}(\,\boldsymbol{\theta}_j \,|\, \mathcal{S}_{\mathrm{bn}}\,) \right).
\end{aligned}$$

Hierdoor kunnen deze parameters ook op een lokale wijze geleerd en gebruikt worden.

Hoe deze verdeling er in de praktijk uitziet, hangt af van de conditionele verdeling die men kiest. Een veel voorkomende keuze is de tabelverdeling. Hier gebruiken we een *verschillende* tabelverdeling voor elke variabele *en* oudercombinatie. We noteren met $\boldsymbol{\theta}_j$ de parameters van de verdeling van $\underline{x}_j$ geconditioneerd op zijn ouders π. Dit is de verzameling van tabelparameters $\boldsymbol{\theta}_{j,\pi_k}$ voor elke verschillende ouderconfiguratie $\pi(\underline{x}_j) = \pi_k$:

$$\mathrm{p}(\,\underline{x}_j = m \,|\, \pi(\underline{x}_j) = \pi_k, \boldsymbol{\theta}_j\,) = \boldsymbol{\theta}^m_{j,\pi_k}.$$

Een belangrijke eigenschap van deze tabelverdeling is dat de parameters van deze verdeling *kansen* zijn. Aan deze parameters kunnen we dus een duidelijke en intuïtieve betekenis hechten wat ons zal toelaten om achtergrondinformatie in rekening te nemen. Ook is het belangrijk op te merken dat we *veel* parameters introduceren door een aparte tabelverdeling te nemen voor elke verschillende ouderconfiguratie. Deze overvloed aan parameters zal het leren op basis van een dataset bemoeilijken.

We plaatsen deze tabelverdeling in het Bayesiaanse denkkader, en kiezen als hyperverdeling een Dirichlet verdeling vanuit theoretische overwegingen [41]:

$$\begin{aligned}
\mathrm{p}(\,\boldsymbol{\theta}_{j,\pi_k} \,|\, m_1, \ldots, m_d\,) &= \frac{\Gamma(\sum_l m_l)}{\prod_l \Gamma(m_l)} \prod_l (\boldsymbol{\theta}^l_{j,\pi_k})^{m_l - 1} \quad \text{met} \\
\sum_l \boldsymbol{\theta}^l_{j,\pi_k} &= 1, \\
\mathrm{E}[\boldsymbol{\theta}_{j,\pi_k}] &= \left(\frac{m_1}{\sum_l m_l}, \frac{m_2}{\sum_l m_l}, \ldots, \frac{m_d}{\sum_l m_l} \right), \\
\mathrm{V}[\boldsymbol{\theta}^l_{j,\pi_k}] &= \frac{m_l(1 - m_l / \sum_r m_r)}{(\sum_r m_r + 1) \sum_r m_r}.
\end{aligned}$$

De parameters $(m_1, \ldots, m_d)$ worden ook wel pseudocounts genoemd vanwege de directe interpretatie als aantal datarecords dat deze prior waard is. Verder is deze verdeling toegevoegd aan de tabelverdeling en daarom ook handig om mee te werken. Dit laat ons bijvoorbeeld toe om zowel de kans van een lokale substructuur als de a posteriori parameterverdeling exact neer te schrijven:

$$\begin{aligned}
\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}}\,|\,\mathbf{D}\,) &\propto \prod_{j=1}^{v} L(\,j\,|\,\pi_j\,) \\
&= \prod_{j=1}^{v} \int_{\boldsymbol{\Theta}_j} \prod_{i=1}^{n} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i), \mathcal{S}_{\mathrm{bn}}, \boldsymbol{\theta}_j\,)\,\mathrm{p}(\boldsymbol{\theta}_j)\,d\boldsymbol{\theta}_j \\
&= \prod_{j=1}^{v} \int_{\boldsymbol{\Theta}_j} \prod_{i=1}^{n} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i), \mathcal{S}_{\mathrm{bn}}, \boldsymbol{\theta}_j\,) \prod_{k=1}^{q_j} \mathrm{p}(\boldsymbol{\theta}_{j,\pi_k})\,d\boldsymbol{\theta}_j \\
&= \prod_{j=1}^{v} \int_{\boldsymbol{\Theta}_j} \prod_{k=1}^{q_j} \frac{\Gamma(\sum_l m_{jkl})}{\prod_l \Gamma(m_{jkl})} \prod_{l=1}^{d} (\boldsymbol{\theta}_{j,\pi_k}^l)^{n_{jkl}+m_{jkl}-1} d\boldsymbol{\theta}_j \\
&= \prod_{j=1}^{v} \prod_{k=1}^{q_j} \frac{\Gamma(\sum_l m_{jkl})}{\prod_l \Gamma(m_{jkl})} \frac{\prod_l \Gamma(n_{jkl}+m_{jkl})}{\Gamma(\sum_l n_{jkl} + \sum_l m_{jkl})}.
\end{aligned}$$

Hierbij is n_{jkl} het aantal datarecords dat we observeren in de dataset $\mathbf{D}$ waarbij $\underline{x}_j = l$ en de ouders van $\underline{x}_j$ de ouderconfiguratie π_k aannemen. Met q_j duiden we het aantal verschillende oudercombinaties voor $\underline{x}_j$ aan.

Dit geeft ons de volgende formule voor de kans van een lokale substructuur:

$$L(\,j\,|\,\pi_j\,) = \prod_{k=1}^{q_j} \frac{\Gamma(\sum_l m_{jkl})}{\prod_l \Gamma(m_{jkl})} \frac{\prod_l \Gamma(n_{jkl}+m_{jkl})}{\Gamma(\sum_l n_{jkl} + \sum_l m_{jkl})}.$$

In bovenstaande formule kunnen we de factor

$$\frac{\Gamma(\sum_l m_{jkl})}{\prod_l \Gamma(m_{jkl})}$$

weglaten als we de pseudocounts m_{jkl} onafhankelijk van de ouders of het *aantal* ouders kiezen.

De a posteriori parameterverdeling blijft, net als de a priori parameterverdeling, een product van Dirichlet verdelingen. Enkel de *parameters* van deze verdelingen worden aangepast op basis van de statistieken n_{jkl} van de dataset:

$$\begin{aligned}
\mathrm{p}(\,\boldsymbol{\theta}\,|\,\mathbf{D}, \mathcal{S}_{\mathrm{bn}}\,) &\propto \mathcal{L}(\,\boldsymbol{\theta}\,|\,\mathbf{D}\,)\,\mathrm{p}(\boldsymbol{\theta}) \\
&= \prod_{j=1}^{v} \prod_{i=1}^{n} \boldsymbol{\theta}_{j,\pi(x_j^i)}^{x_j^i}\,\mathrm{p}(\boldsymbol{\theta}_j) \\
&= \prod_{j=1}^{v} \prod_{k=1}^{q_j} \prod_{l=1}^{d} (\boldsymbol{\theta}_{j,\pi_k}^l)^{n_{jkl}}\,\mathrm{p}(\boldsymbol{\theta}_{j,\pi_k}^l) \\
&= \prod_{j=1}^{v} \prod_{k=1}^{q_j} \prod_{l=1}^{d} (\boldsymbol{\theta}_{j,\pi_k}^l)^{n_{jkl}+m_{jkl}-1}.
\end{aligned}$$

Overige algoritmen

Eens we een Bayesiaans netwerk gespecificeerd hebben met een structuur en parameters, kunnen we het gaan gebruiken om voorspellingen mee te doen. Het berekenen van de benodigde conditionele en marginale verdelingen kan gedaan worden met behulp van het “probability propagation in tree of cliques” algoritme [45].

Om het Bayesiaans netwerk te gebruiken in de transformatiesetup uit de vorige sectie, willen we ook efficiënt random vectoren genereren volgens de gezamelijke verdeling dat ons netwerk voorstelt. Een Bayesiaans netwerk is echter gebaseerd op de kettingregel voor kansen. Hierdoor kunnen we rechtstreeks random vectoren aanmaken door in de vorlgorde van de variabelen een waarde voor elke veranderlijke te genereren op basis van díe conditionele verdeling die overeenkomt met de reeds gegenereerde waarden.

Het verzamelen van achtergrondinformatie

Om efficiënt gebruik te maken van Bayesiaanse netwerken om achtergrondinformatie te verzamelen, moeten we deze informatie hiermee op een eenvoudige wijze kunnen voorstellen. De opsplitsing van een Bayesiaans netwerk in een structuur en bijhorende parameters en het gebruik van de tabelverdeling, biedt ons een aantal mogelijkheden.

Expertkennis

Voor het tumorprobleem kon Prof. Timmerman zijn kennis en ervaring neerschrijven met behulp van een Bayesiaans netwerk met een vaste structuur en parameters. Dit was echter enkel mogelijk door het aantal variabelen te restricteren tot 11. De structuur van dit model is weergegeven in Figuur 7.1. Deze structuur werd gradueel opgebouwd en Prof. Timmerman specificeerde de parameters van dit model door een vragenlijst van 29 bladzijden in te vullen, met vragen als:

> Beschouw een patiënt met een *kwaadaardige* tumor, de aanwezigheid van *ascites* en *premenopausaal*. Wat is de kans dat $CA125 < 35$?

Omdat het rechtstreeks specificeren van getallen niet eenvoudig is, gaven we de mogelijkheid om op een grafische wijze antwoord te geven door een aanduiding te maken op een schaal van 0 tot 1. Deze schaal bevatte aanduidingen als *onmogelijk*, *onwaarschijnlijk*, *onzeker*, *fifty-fifty*, *verwacht*, *waarschijnlijk* en *zeker*.

Bij het specificeren en interpreteren van netwerkstructuren is het belangrijk om deze structuren op een overzichtelijke wijze voor te stellen. Dit houdt de visualisatie in van het netwerk in twee dimensies waarbij de knopen zo uniform mogelijk verspreid zijn terwijl onderling verbonden knopen toch zo dicht mogelijk bij elkaar liggen. Deze eis deed ons denken aan het “self organizing map”-algoritme (SOM) ontwikkeld door Kohonen [53]. Dit algoritme heeft tot doel een hoogdimensionale dataset te visualiseren in een lagere dimensie. Typisch wordt een tweedimensionale gridstructuur van neuronen gebruikt waarbij het algoritme de dataverdeling probeert te benaderen door neuronen te plaatsen in die gebieden waar veel datapunten voorkomen, onder de beperking van de grid netwerkstructuur die verbonden neuronen dicht bij elkaar houdt.

We pasten dit algoritme "omgekeerd" toe door een tweedimensionale uniform verdeelde dataset aan te maken en het SOM-algoritme toe te passen met de te visualiseren netwerkstructuur. Dit SOM-algoritme probeert nu de dataverdeling te benaderen door het plaatsen van de knopen waarbij het verbonden knopen toch dicht bij elkaar wil houden. We krijgen dus de gewenste uniforme spreiding van de knopen waarbij verbonden knopen toch dicht bij elkaar liggen. Bij wijze van voorbeeld toont Figuur 5.5 een netwerk waarbij de plaatsing van de knopen random gebeurde, terwijl ditzelfde netwerk na toepassing van het SOM-algoritme getoond wordt in Figuur 5.6.

Similariteitsinformatie

Verder kon Prof. Timmerman ons extra informatie verschaffen over de onderlinge samenhang van *alle* variabelen. Deze informatie bestaat uit een numerieke waarde die de sterkte van de paarsgewijze verbanden tussen de variabelen aangeeft. Deze paarsgewijze verbanden $\boldsymbol{V}_{\underline{x}\underline{y}}$ kunnen we transformeren naar een *a priori* verdeling over de ruimte van Bayesiaanse netwerken, uitgaande van de assumptie dat de a priori kans van een netwerkstructuur gefactoriseerd kan worden in afzonderlijke *pijlkansen*:

$$\begin{aligned} \mathrm{p}(\mathcal{S}_{\mathrm{bn}}) &= \prod_{\underline{x}} \mathrm{p}(\pi(\underline{x}) \to \underline{x}) \\ \mathrm{p}(\pi(\underline{x}) \to \underline{x}) &= \prod_{\underline{y}\in\pi(\underline{x})} \mathrm{p}(\underline{y} \to \underline{x}) \prod_{\underline{y}\notin\pi(\underline{x})} (1 - \mathrm{p}(\underline{y} \to \underline{x})). \end{aligned}$$

We definiëren de kans om een pijl van $\underline{y}$ naar $\underline{x}$ te observeren als

$$\mathrm{p}(\underline{y} \to \underline{x}) = \boldsymbol{V}^{\zeta}_{\underline{x}\underline{y}},$$

hetgeen ons onmiddellijk ook de kans geeft om geen pijl te observeren.

Hierbij biedt ζ ons controle over het verwachte aantal ouders per variabele, zoals aangegeven in Sectie 5.5.1.

Tekstuele informatie

Als laatste onderzochten we een techniek om eveneens een verdeling over de netwerkstructuren te definiëren, ditmaal gebaseerd op de medische literatuur. We wensen dit op een geautomatiseerde wijze te doen omdat het effectief *lezen* van al deze documenten een onmogelijke taak is.

Veel wetenschappelijke documenten zijn immers gepubliceerd over het domein van ovariale tumoren en we gaan ervan uit dat de meeste documenten op een *positieve* manier verbanden bespreken; wanneer verschillende variabelen besproken worden in een document wil men meestal aantonen dat er een verband *is* tussen deze, in plaats van aantonen dat ze net *geen* verband met elkaar houden.

Onze techniek bepaalt het verband tussen $\underline{x}$ en $\underline{y}$ aan de hand van het aantal documenten dat tezamen handelt over $\underline{x}$ en $\underline{y}$. Hiertoe converteren we eerst elk document $\mathcal{D}$ naar een vectorrepresentatie $\mathcal{T}(\mathcal{D})$; elke component van deze vector geeft het gewicht aan van een bepaald woord in dat document. Het gewicht van een woord $\circledast$ binnen het document $\mathcal{D}$ bepalen we met behulp van de *woord-frequentie inverse-document-frequentie* (tf-idf):

$$\omega(\circledast \mid \mathcal{D}) = -\frac{\#\circledast \in \mathcal{D}}{\#\mathcal{D}} \log\left(\frac{\#\mathcal{C}}{\#\mathcal{C}|_{\circledast}}\right).$$

In deze formule stelt $\frac{\#\circledast\in\mathcal{D}}{\#\mathcal{D}}$ de frequentie van het woord $\circledast$ in het document $\mathcal{D}$ voor terwijl de factor $-\log\left(\frac{\#\mathcal{C}}{\#\mathcal{C}|_{\circledast}}\right)$ de logaritme van de frequentie van documenten die $\circledast$ bevatten, voorstelt. Deze laatste factor elimineert frequente woorden zoals "the", "for" of "a".

Verder gaan we ervan uit dat elke variabele geannoteerd is. Voor het ovariale tumorprobleem bestaat deze annotatie uit definities uit medische woordenboeken, de omschrijvingen van deze variabelen in het IOTA protocol, de omschrijving uit de doctoraatsthesis van Prof. Timmerman [81] en relevante medische artikels.

De annotatie van elk van deze variabelen converteren we naar de vectorrepresentatie met behulp van bovenstaande techniek. Deze kunnen we op twee manieren gebruiken: als eerste kunnen we met deze vectorrepresentaties *rechtstreeks* een similariteitsmaat tussen de variabelen definiëren. Een veelgebruikte maat is de cosinus van de hoek tussen deze twee vectoren:

$$\boldsymbol{V}_{\underline{x}\underline{y}} = \boldsymbol{V}_{\underline{y}\underline{x}} = \mathrm{sim}(\mathcal{D}(\underline{x}),\mathcal{D}(\underline{y})) = \frac{<\mathcal{T}(\mathcal{D}(\underline{x})),\mathcal{T}(\mathcal{D}(\underline{y}))>}{\|\mathcal{T}(\mathcal{D}(\underline{x}))\|\|\mathcal{T}(\mathcal{D}(\underline{y}))\|}.$$

Een tweede mogelijkheid definieert het verband tussen twee variabelen op basis van hun gezamelijk voorkomen in een corpus $\mathcal{C}$ van documenten:

$$\boldsymbol{V}_{\underline{x}\underline{y}} = \boldsymbol{V}_{\underline{y}\underline{x}} = \mathrm{P}(\,\underline{x}\in\mathcal{D} \text{ en } \underline{y}\in\mathcal{D} \,|\, \underline{x}\in\mathcal{D} \text{ of } \underline{y}\in\mathcal{D}, \mathcal{D}\in\mathcal{C}\,).$$

De notatie $\underline{x} \in \mathcal{D}$ duidt aan dat document $\mathcal{D}$ handelt over de variabele $\underline{x}$, hetgeen we detecteren als de vectorsimilariteit $\mathrm{sim}(\mathcal{T}(\underline{x}),\mathcal{T}(\mathcal{D}))$ boven een bepaalde drempelwaarde uitkomt.

Merk op dat in de praktijk een aantal preprocessing stappen nodig zijn zoals het converteren van elk woord naar zijn stamvorm of het afhandelen van domeinspecifieke woorden, phrasen of synoniemen.

Meerlaagse perceptrons

Na het donormodel introduceren we het acceptormodel waarmee we wensen te classificeren en te leren van de statistische dataset. Verder zullen we voor dit acceptormodel ook een informatieve a priori verdeling definiëren met behulp van de transformatie methode en het donormodel.

We kozen voor het *meerlaagse perceptron* vanwege zijn goede leereigenschappen en omdat het natuurlijk omspringt met continue data. Verder modelleert het specifiek een invoer-uitvoer functie in plaats van een gezamelijke verdeling zoals het geval was bij Bayesiaanse netwerken. De mogelijkheden om achtergrondinformatie mee in rekening te nemen zijn echter beperkt, hetgeen we willen oplossen met behulp van de transformatie methode die hierboven besproken werd.

Voorwaardelijke verdelingen

Een meerlaags perceptron is een geparametriseerde functie van $\mathbb{R}^d$ naar $\mathbb{R}$. Deze functie is opgebouwd uit kleine modulen — *perceptrons* of *neuronen* genaamd — die laag per laag met elkaar zijn verbonden (zie Figuur 6.1). Deze kleine modulen zijn geïnspireerd op de neuronen uit de hersenen wat ook hun naam verklaart. Elk perceptron berekent de gewogen som van de uitvoeren van de perceptrons van de vorige laag en telt hier een bias bij op, stuurt vervolgens deze waarde door een *doorlaat*functie $\phi(\cdot)$ en geeft deze waarde tenslotte door aan de volgende laag:

$$\mathrm{mlp}(\,\boldsymbol{x} \,|\, \boldsymbol{\omega}\,) = \phi(\ldots\phi(\sum_j \phi(\sum_i x_i \boldsymbol{\omega}^1_{ji} + \boldsymbol{\omega}^1_{jb})\boldsymbol{\omega}^2_{kj} + \boldsymbol{\omega}^2_{kb})\ldots).$$

In deze thesis zullen we enkel gebruik maken van de tangent hyperbolicus doorlaatfunctie (zie Figuur 6.2) of de lineaire doorlaatfunctie.

Regressie

We kunnen dit meerlaagse preceptron gebruiken in de regressiecontext. Dit houdt in dat we de uitvoer van het neurale netwerk gebruiken als een model voor het gemiddelde van $\underline{y}$, geconditioneerd op de invoer $\underline{\boldsymbol{x}}$:

$$\mathrm{E}[\underline{y} \mid \underline{\boldsymbol{x}}] = \mathrm{mlp}(\,\underline{\boldsymbol{x}} \,|\, \boldsymbol{\omega}\,).$$

Met behulp van dit voorwaardelijke gemiddelde en een verdeling hierrond definiëren we een voorwaardelijke verdeling. Een veelgebruikte keuze is een Gaussiaanse verdeling met een constante variantie σ^2 onafhankelijk van $\underline{\boldsymbol{x}}$:

$$\mathrm{p}(\,\underline{y} \,|\, \underline{\boldsymbol{x}}, \boldsymbol{\omega}\,) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(\underline{y} - \mathrm{mlp}(\,\underline{\boldsymbol{x}} \,|\, \boldsymbol{\omega}\,))^2}.$$

Logistieke regressie

Een andere mogelijkheid, vooral toegepast bij classificatie, bestaat erin de uitvoer van het netwerk te interpreteren als de kans dat een bepaald record met observaties $\underline{\boldsymbol{x}}$ tot de klasse $\mathcal{C}_\mathrm{P}$ behoort:

$$\mathrm{P}(\,\underline{y} = \mathcal{C}_\mathrm{P} \,|\, \underline{\boldsymbol{x}}, \boldsymbol{\omega}\,) = \frac{1 + \mathrm{mlp}(\,\underline{\boldsymbol{x}} \,|\, \boldsymbol{\omega}\,)}{2}.$$

Hierbij transformeren we het bereik $]-1,1[$ van het neurale netwerk op een lineaire wijze tot $]0,1[$ zodat we de uitvoer onmiddellijk kunnen interpreteren als een kans. Wegens symmetrieredenen maken we liever geen gebruik van de sigmoïdale doorlaatfunctie die wel het bereik $]0,1[$ heeft.

Het gebruik van de tangent hyperbolicus doorlaatfunctie is gemotiveerd vanuit het *logistieke regressie* oogpunt. Hier modelleert men de logaritme van de kansverhouding van $\mathrm{P}(\underline{y}=1)/(1-\mathrm{P}(\underline{y}=1))$ met een lineair model

$$\begin{aligned}\log\left(\frac{\mathrm{P}(\underline{y}=1)}{1-\mathrm{P}(\underline{y}=1)}\right) &= \beta_0+\beta_1\underline{x}_1+\cdots+\beta_v\underline{x}_v \\ &= \mu,\end{aligned}$$

waaruit men haalt dat

$$\begin{aligned}\mathrm{P}(\underline{y}=1) &= \frac{e^{\mu}}{1+e^{\mu}} \\ &= \frac{1}{1+e^{-\mu}} \\ &= \mathrm{sigmoid}(\mu) \\ &= \frac{1+\tanh(\mu)}{2}.\end{aligned}$$

Het logistieke regressiemodel is analoog aan een eenvoudig neuraal netwerk dat bestaat uit slechts één perceptron met een tangent hyperbolicus doorlaatfunctie. Op gelijkaardige wijze wordt een lineair regressiemodel voorgesteld door een neuraal netwerk met één neuron, een lineaire doorlaatfunctie en een Gaussiaanse verdeling rond dit voorwaardelijke gemiddelde. Door extra lagen van neuronen toe te voegen aan dit neurale netwerk, kunnen we gradueel de complexiteit van dit voorwaardelijke gemiddelde verhogen en niet-lineaire voorwaardelijke gemiddelden of beslissingsgrenzen implementeren.

Elke continue functie kan immers willekeurig goed benaderd worden met een meerlaags perceptron als we voldoende neuronen nemen [42]. Verder leert de praktijk ons dat een redelijke benadering in de meeste gevallen al met een beperkt aantal neuronen bereikt wordt, hetgeen het aantal parameters beperkt en de leereigenschappen ten goede komt.

A posteriori verdeling

Nu we weten hoe we een verdeling voorstellen met behulp van een meerlaags perceptron, gaan we hiermee leren. Hoewel het gros van de literatuur over het leren van neurale netwerken gebaseerd is op het minimaliseren van kostfuncties met eventueel bijgevoegde complexiteitstermen, formuleren we het probleem in het kader van de kansrekening. Dit is volledig analoog is aan het klassieke kostfunctiekader:

$$\begin{aligned}
\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) &= \frac{\mathrm{p}(\,\mathbf{D}\,|\,\boldsymbol{\omega}\,)\,\mathrm{p}(\boldsymbol{\omega})}{\mathrm{p}(\mathbf{D})} \\
&\propto \mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)\,\mathrm{p}(\boldsymbol{\omega}) \\
&= \prod_{i=1}^{n} \mathrm{p}(\,\underline{y}_i\,|\,\underline{\boldsymbol{x}}_i, \boldsymbol{\omega}\,)\,\mathrm{p}(\,\underline{\boldsymbol{x}}_i\,|\,\boldsymbol{\omega}\,)\,\mathrm{p}(\boldsymbol{\omega}) \\
&\propto \prod_{i=1}^{n} \mathrm{p}(\,\underline{y}_i\,|\,\underline{\boldsymbol{x}}_i, \boldsymbol{\omega}\,)\,\mathrm{p}(\boldsymbol{\omega}) \\
\mathrm{Error}^*(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) &= -\log(\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)) \\
&= \mathrm{Error}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) - \log(\mathrm{p}(\boldsymbol{\omega})) + \mathrm{C}^{\underline{\mathrm{te}}}. \\
&= \mathrm{Error}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) + \frac{1}{2\sigma_{\mathrm{wd}}^2}\sum_i \boldsymbol{\omega}_i^2 + \mathrm{C}^{\underline{\mathrm{te}}}.
\end{aligned}$$

Meestal kiest men een Gaussiaanse a priori verdeling $\mathrm{p}(\boldsymbol{\omega}) \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathrm{wd}}^2)$ om de norm van de parameters beperkt te houden, hetgeen overeenstemt met de extra som van gekwadrateerde gewichten in de laatste formule. Deze term wordt ook wel een *weight decay* regularizatieterm genoemd.

Verder behandelen we meerlaagse perceptrons ook op een Bayesiaanse wijze waarbij we uitmiddelen over de a posteriori verdeling in plaats van de *maximum a posteriori* parametrisatie te zoeken:

$$\begin{aligned}
\mathrm{P}(\,\underline{y} = \mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x}, \mathbf{D}, \xi\,) &= \int_{\boldsymbol{\Omega}} \mathrm{P}(\,\underline{y} = \mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x}, \boldsymbol{\omega}\,)\,\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}, \xi\,)\,d\boldsymbol{\omega} \\
&\approx \frac{1}{N}\sum_{i=1}^{N} \mathrm{P}(\,\underline{y} = \mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x}, \boldsymbol{\omega}_i\,) \ \text{ met } \ \boldsymbol{\omega}_i \sim \mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}, \xi\,) \\
&\neq \mathrm{P}(\,\underline{y} = \mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x}, \boldsymbol{\omega}^*\,) \ \text{ met } \ \boldsymbol{\omega}^* = \mathrm{argmax}_{\boldsymbol{\omega}}(\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}, \xi\,)).
\end{aligned}$$

We benaderen de integraal met behulp van een Monte Carlo som gebaseerd op de a posteriori parameterverdeling $\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}, \xi\,)$. De nodige toevalsvectoren volgens deze a posteriori verdeling genereren we met behulp van de hybride Monte Carlo Markov keten methode. Deze implementeert een Markov keten waarbij nieuwe toestanden gegenereerd worden door het volgen van de vergelijkingen van Hamilton van een imaginair fysisch systeem met als Boltzmannverdeling de gewenste a posteriori verdeling $\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}, \xi\,)$ (zie Sectie 6.7.2 voor de details).

Informatieve a priori verdeling

Nu we neurale netwerken kunnen gebruiken in het Bayesiaanse kader onderzoeken we de mogelijkheden om a priori kennis mee in rekening te nemen. Om dit te verwezenlijken moeten we een a priori parameterverdeling specificeren die de achtergrondinformatie bevat. Dit gaat echter niet rechtoe rechtaan omdat we geen duidelijke betekenis kunnen hechten aan de parameters van een neuraal netwerk, terwijl dit voor een Bayesiaans netwerk gebaseerd op de tabelverdeling wel het geval was. Daar waren de parameters immers kansen.

Zoals reeds eerder aangeduid zullen we de achtergrondinformatie omschrijven met behulp van een Bayesiaans netwerk. Deze verdeling transformeren we vervolgens naar de gewichtsruimte van het meerlaagse perceptron door middel van *virtuele datasets*.

Hiertoe genereren we verschillende parametervectoren volgens de *informatieve a priori* verdeling $\mathrm{p}(\boldsymbol{\omega} \mid \xi)$ met behulp van het algoritme dat in de transformatiesectie beschreven staat. Vervolgens gebruiken we deze verzameling van parametervectoren $\{\boldsymbol{\omega}_1, \ldots \boldsymbol{\omega}_d\}$ om de verdeling $\mathrm{p}(\boldsymbol{\omega} \mid \xi)$ te schatten met behulp van een parametrische verdeling $\mathrm{p}(\boldsymbol{\omega} \mid \boldsymbol{\nu})$, zoals een multivariate Gaussiaanse verdeling.

Verdelingen in de gewichtsruimte en symmetrieën

Het schatten van deze verdeling wordt echter bemoeilijkt door de aanwezigheid van symmetrieën in de gewichtsruimte van een neuraal netwerk; er bestaan *verschillende* parametrisaties $\boldsymbol{\omega}_1 \neq \boldsymbol{\omega}_2$ die resulteren in exact dezelfde invoer-uitvoer mapping

$$\mathrm{mlp}(\boldsymbol{x} \mid \boldsymbol{\omega}_1) = \mathrm{mlp}(\boldsymbol{x} \mid \boldsymbol{\omega}_2),\ \forall\, \boldsymbol{x},$$

en dus in dezelfde voorwaardelijke verdeling.

Er zijn twee verschillende symmetrieën die optreden: een eerste type wordt veroorzaakt door het permuteren van neuronen in een verborgen laag. Hierdoor zullen de parametrisatievectoren op gelijkaardige wijze gepermuteerd worden, maar de netwerkfunctie wordt niet gewijzigd.

Een tweede type van symmetrie wordt veroorzaakt door het gebruik van oneven functies; door alle gewichten die een bepaald neuron in- en uitkomen van teken te wisselen, blijft de globale netwerkfunctie behouden, maar krijgen we een andere parametrisatie. Omdat enkel het teken wijzigt, blijft de norm ongewijzigd; deze verschillende symmetrische voorstellingen van dezelfde functie zullen op dezelfde wijze gepenalizeerd worden door een weight decay term of de equivalente Gaussiaanse a priori verdeling. Het gebruik van de sigmoïdale doorlaatfunctie resulteert ook in een symmetrie die echter de norm wel wijzigt, en dus ook de a priori kans volgens zo'n complexiteitsgebaseerde verdeling. Dit is onze voornaamste reden om de tangent hyperbolicus doorlaatfunctie te prefereren boven de sigmoïdale doorlaatfunctie.

De verdeling $\mathrm{p}(\boldsymbol{\omega} \mid \mathbf{D}, \xi)$ zal eveneens deze symmetrieën vertonen. We wensen echter maar één van deze modes te schatten door elke gegenereerde parametrisatie $\boldsymbol{\omega}_i$ te transformeren naar een bepaald quadrant door een transformatie $T_i(\cdot)$. Deze transformatie is de combinatie van een permutatie van de knopen in elke verborgen laag en het eventueel omwisselen van de tekens per knoop. Dit leert ons dat het aantal symmetrieën gelijk is aan

$$\#\text{Symmetrieën} = \prod_k 2^{N_k} N_k!,$$

waarbij k over de verborgen knopen loopt en N_k het aantal neuronen in de k^{de} laag voorstelt.

We wensen nu díe transformaties $\{T_i\}_{i=1}^d$ te vinden zodat we $\{T(\boldsymbol{\omega}_i)\}_{i=1}^d$ zo goed mogelijk kunnen schatten:

$$\begin{aligned} \{T_i(\cdot)\}_{i=1}^d &= \mathrm{argmax}_{T_i}(\max_{\boldsymbol{\nu}} \mathrm{p}(\,\{T_i(\boldsymbol{\omega}_i)\}_{i=1}^d \,|\, \boldsymbol{\nu}\,)) \\ &= \mathrm{argmax}_{T_i}(\max_{\boldsymbol{\nu}} \mathcal{L}(\,\boldsymbol{\nu} \,|\, \{T_i(\boldsymbol{\omega}_i)\}_{i=1}^d\,)). \end{aligned}$$

Deze transformaties op een exhaustatieve wijze berekenen is onmogelijk, vermits dit #Symmetrieën^d optimizaties van de likelihood vergt. Om het schatten van deze verdeling toch mogelijk te maken, ontwikkelden we een techniek die de exponent d omvormt naar een multiplicatieve factor d door toepassing van het EM algoritme op een tegen-intuïtieve wijze (zie Sectie 6.9.2 voor de details)

$$\#\text{Symmetrieën}^d \to d\,\#\text{Symmetrieën}$$

In plaats van alle transformaties *tezamen* te zoeken, kunnen we nu de transformatie voor elke parametervector *afzonderlijk* zoeken.

Verder ontwikkelden we een bijkomende heuristiek om het zoeken naar één transformatie sneller te maken. Deze heuristiek postuleert per verborgen laag een volgorde voor de knopen, gebaseerd op hun belangrijkheid. Deze belangrijkheid wordt gemeten aan de hand van de L_2 norm van de vector van gewichten verbonden aan elke knoop. Deze volgorde wordt gebruikt als uitgangspositie en wordt aangepast op basis van de volledige informatie van de gewichten, zoals uitgelegd in meer detail in Sectie 6.9.2.

Een voorbeeld

Het gedrag van deze heuristiek wordt geïllustreerd op een artificieel voorbeeld. Een sinusdataset met ruis definieert een a posteriori verdeling. Met behulp van het hybride Monte Carlo Markov keten algoritme, genereerden we 100 netwerkparametrisaties volgens deze a posteriori verdeling. De data, samen met deze 100 gegenereerde netwerk*functies*, wordt getoond in Figuur 6.15. De gegenereerde netwerk*parametrisaties* worden getoond in Figuur 6.16. Hier stelt elke kolom één parametrisatie voor en geeft de grijswaarde een indicatie van de waarde van elk gewicht aan. Gelijkaardige gewichten over de verschillende parametrisaties — de rijen — nemen sterk verschillende waarden aan hetgeen hun schatting sterk bemoeilijkt. Figuur 6.18 toont dezelfde parametrisatievectoren *nadat* we met bovenstaande methode de gepaste transformaties hebben gezocht. De variatie per rij is drastisch verminderd, en we verwachten dan ook dat we deze verdeling beter zullen kunnen schatten.

We schatten de parameters van een multivariate Gaussiaanse verdeling op zowel de originele netwerkparametrisaties als die na onze transformatie. De netwerkfuncties gedefinieerd door de respectievelijke gemiddeldenvector van deze verdelingen worden getoond in Figuur 6.20. De gestreepte lijn is afkomstig van de schatting op basis van de originele vectoren. Deze is de nulfunctie, hetgeen we verwachten; de originele parametervectoren liggen immers random maar evenwichtig verdeeld rond de oorsprong door de symmetrieën. De individuele netwerkfuncties volgen de sinustrend mooi maar de geschatte gemiddeldenvector is ongeveer de nulvector, wat resulteert in de nul netwerkfunctie.

De volle lijn in de figuur is de netwerkfunctie afkomstig van het gemiddelde *nadat* we de gepaste transformaties hebben gezocht die onze netwerkparametrisaties mooi bijeenbrengt. Deze functie volgt wél mooi de sinustrend wat aantoont dat we één mode hebben geschat. De stippellijn is afkomstig van een meer primitieve methode en werkt slechts half, zoals uitgelegd in Sectie 6.9.2.

Nu we onze informatieve a priori verdeling kunnen schatten, die gedefinieerd is op basis van een Bayesiaans netwerk en virtuele datasets, updaten we deze analytische a priori verdeling naar de a posteriori verdeling op basis van de echte dataset met behulp van het hybride Monte Carlo Markov keten algoritme. Met deze Markov keten kunnen we tenslotte de kans bepalen dat een bepaalde tumor kwaadaardig is, op basis van de observatievector $\boldsymbol{x}$.

Classificatie van ovariale tumoren

Nu alle technieken geïntroduceerd zijn passen we ze toe ovariale tumorprobleem dat een aantal secties geleden besproken werd. We brengen even ter herinnering dat we preoperatief de kans willen bepalen dat een bepaalde tumor met observatievector $\boldsymbol{x}$ kwaadaardig is. We zullen het leergedrag van een uiteenlopende reeks modellen onderling vergelijken en uitzetten tegenover vorige studies omtrent dit probleem.

Informatiebronnen

Vooraleer we van start gaan, nemen we de dataset onder de loep. Voor de Bayesiaanse netwerken discretiseren we de continue variabelen met behulp van discretisatie-intervallen die Prof. Timmerman specificeerde. Deze staan beschreven in Appendix A. Voor de neurale netwerken transformeerden we sommige variabelen, introduceerden we een aantal design variabelen en normaliseerden we de variabelen zodat het neurale netwerk grootheden binnenkrijgt van grosso modo dezelfde grootte orde. De details zijn terug te vinden in Sectie 7.2.1.

Vervolgens vergeleken we de a priori verdelingen over de ruimte van Bayesiaanse netwerkstructuren gedefinieerd op basis van de expertinformatie, de tekstuele informatie volgens de directe annotatiesimilariteit of gebaseerde op een groot aantal domein documenten, als op de dataset als referentie. We visualiseerden deze paarsgewijze verbanden respectievelijk in de Figuren 7.2, 7.5, 7.6 en 7.7, waar we de sterkte van een verband aangeven door middel van de grijswaarde en de dikte van de pijlen. We zien dat deze verbanden grosso modo overeen komen met die gevonden in de dataset, dit toont aan dat we op een succesvolle wijze deze similariteitsinformatie hebben verzameld.

De leercurven

We zijn nu klaar om het leergedrag van een uiteenlopende reeks modellen te vergelijken. Zoals eerder aangegeven, zullen we de oppervlakte onder de ROC curve gebruiken als performantiemaat. Op basis van deze maat gaan we een *leercurve* berekenen welke het leergedrag van een bepaald model zal visualiseren. Zo'n leercurve zet de gemiddelde oppervlakte onder de ROC curve uit in functie van het percentage datarecords dat we gebruiken om de a posteriori verdeling mee te berekenen. Op de x as zetten we dit percentage trainingrecords uit, terwijl we op de y as het gemiddelde uitzetten van de oppervlakte onder de ROC curve, berekend op basis van 1000 tweevoudige crossvalidatiesessies met een overeenstemmend percentage trainingrecords. De trainingrecords worden gebruikt om de a posteriori verdeling te specificeren terwijl we de overige records gebruiken om de ROC performantie te berekenen.

Eenvoudige modellen

Als eerste presenteren we de performantie van een aantal eenvoudige modellen die ons ook de variabelen beter zullen leren kennen. Zo toont Tabel 7.9 de performantie van een aantal univariate modellen. Hier wordt elke variabele *afzonderlijk* gebruikt als classificator, hetgeen ons een ondergrens voor de verwachtte performantie geeft (0.8323409) en aantoont welke variabelen goed gebruikt kunnen worden om de kwaadaardigheid van een ovariale tumor te voorspellen.

Een veelgebruikte vuistregel in de praktijk om een tumor te classificeren is de *Risk of malignancy index* (RMI). Deze regel combineert ultrasonore informatie met de menopausale status en het *CA125* niveau gemeten in het bloed. De performantie van dit model is 0.8891462. Vermits dit model, evenals de bovenstaande univariate modellen, geen parameters bevat, is het zinloos een leercurve hiervoor te berekenen vermits deze modellen niet kunnen bijleren.

Bayesiaanse netwerken

Figuur 7.12 toont de leercurve voor het Bayesiaanse netwerkmodel *zonder* gebruik te maken van enige achtergrondinformatie. Dit houdt in dat de a posteriori structuurverdeling van dit model voor elke crossvalidatiesessie opnieuw berekend wordt zonder gebruik te maken van één van de informatieve structuurverdelingen. Ook de parameter a priori verdeling is niet-informatief. Dit model heeft een performantie rond 0.5 als geen datarecords gebruikt worden om te leren, terwijl deze performantie gaandeweg groeit tot 0.9414.

We zien duidelijk dat het model leert naarmate we meer informatie gebruiken om te leren. Toch willen we dit leergedrag verbeteren op een aantal vlakken. Zo willen we een model ontwikkelen dat *sneller* leert, reeds een *redelijke* performantie heeft zelfs als er nog geen datarecords zijn geobserveerd zijn en een *hogere* performantie bereikt na het observeren van de data.

De eerste eis proberen we te bereiken met behulp van een informatieve structuurverdeling. Dit zorgt ervoor dat de a posteriori structuurverdeling op een meer uitgesproken manier goede netwerkstructuren naar voor schuift.

Figuur 7.14 toont de leercurven voor de Bayesiaanse netwerkmodellen met de expertstructuurverdeling en deze gebaseerd op de tekstuele informatie. De modellen met een informatieve structuurverdeling leren sneller dan het niet-informatieve model. De expertstructuurverdeling presteert hiervan het beste.

Een redelijke performantie nog vooraleer er data geobserveerd is kunnen we bereiken door een informatieve a priori verdeling te definiëren. Hiertoe gebruiken we het vaste structuur netwerk met informatieve parameterverdeling dat Prof. Timmerman gespecificeerd heeft. De leercurve van dit model wordt getoond in Figuur 7.15 door een volle lijn. Dit model heeft inderdaad al een goede performantie *zonder* dat er data geobserveerd werd. Naarmate we meer data gebruiken verhoogt de performantie nog wel hoewel de uiteindelijke performantie minder goed is dan de vorige Bayesiaanse netwerkmodellen. Dit is te wijten aan het beperkte aantal variabelen en de vaste structuur van het netwerk.

Logistieke regressie

We gaan de derde eis proberen in te willigen door gebruik te maken van een andere modelklasse. We beginnen met het bespreken van het logistieke regressiemodel, hetgeen een speciaal geval is van een meerlaags perceptron. Op Figuur 7.16 zien we de leercurve van dit model met een niet-informatieve a priori verdeling. Dit model heeft, zoals verwacht, een performantie rond 0.5 wanneer geen datarecords geobserveerd zijn. De leercurve steigt echter sneller dan de Bayesiaanse netwerkmodellen, met of zonder informatieve structuurverdeling. Dit toont aan dat het Bayesiaanse netwerkmodel inderdaad geen al te goede leereigenschappen heeft.

Vervolgens passen we de ontwikkelde transformatietechniek toe om de informatie van het expertnetwerk met vaste structuur te gebruiken om een informatieve a priori verdeling te schatten voor het logistieke regressiemodel. We maakten gebruik van virtuele datasets met elk 1000 records en genereerde informatieve logistieke regressieparametrisaties op basis van deze virtuele datasets met behulp van het hybride Monte Carlo Markov keten algoritme. Deze verdeling werd geschat met een multivariate Gaussiaanse verdeling. Eens we deze *informatieve* a priori verdeling hebben, berekenen we de leercurve. Hierbij genereren we parametrisaties volgens de a posteriori verdeling met behulp van het hybride Monte Carlo Markov keten algoritme.

Deze leercurve wordt getoond in Figuur 7.17 met een volle lijn, terwijl het Bayesiaanse netwerk met de informatieve parameterverdeling en het niet-informatieve logistieke regressiemodel eveneens getoond worden ter vergelijking. Blijkbaar heeft de transformatie een positief effect gehad; het logistieke regressiemodel met informatieve a priori heeft een betere performantie dan het Bayesiaanse netwerk, beide voordat we data geobserveerd hebben. Dit is een toevalligheid en geen algemene eigenschap van de transformatietechniek. Verder zien we dat dit logistieke regressiemodel eveneens bijleert wanneer data beobserveerd wordt en een gelijkaardige performantie vertoond dan het niet-informatieve logistieke regressiemodel wanneer het gros van de data gebruikt wordt om te leren.

Meerlaagse perceptrons

Een laatste model dat we gaan gebruiken is een neuraal netwerk met één verborgen laag en twee neuronen in deze laag. We berekenen de leercurve van dit model op basis van een niet-informatieve a priori verdeling, en vergelijken de performantie met die van het niet-informatieve logistieke regressiemodel in Figuur 7.18. Beide modellen vertonen hetzelfde leergedrag, terwijl het neurale netwerk een hogere performantie bereikt. Dit is te danken aan zijn flexibelere structuur.

Als laatste passen we de transformatietechniek ook toe voor dit meerlaagse perceptron op dezelfde wijze als voor het logistieke regressiemodel, waarbij we enkel rekening houden met de symmetrieën in de gewichtsruimte. De performantie van dit informatieve meerlaagse perceptronmodel wordt getoond in Figuur 7.19, samen met het informatieve logistieke regressiemodel en het expert Bayesiaans netwerk met vaste structuur en informatieve parameterprior. Hoewel de performantie van het neurale netwerkmodel initieel het minste is van de drie, heeft het toch nog een redelijke performantie. Verder wordt dit al na het observeren van 5% van de data het best presterende model, gevolgd door het logistieke regressiemodel en het Bayesiaanse netwerk.

Implementatie

De bovenstaande modellen en technieken zijn tamelijk computerintensief en een groot aantal verschillende algoritmen moet gecombineerd worden. De implementatie hiervan kon niet op één nacht afgehandeld worden.

Alhoewel er verschillende softwarepakketten bestaan die bepaalde gebruikte algoritmen implementeren, zijn deze vaak te restrictief of ongeschikt voor onze doeleinden. Na stevig wat nadenkwerk besloten we om al de algoritmen en modellen volledig “from scratch” te implementeren. Hoewel dit op het eerste gezicht een vreemde keuze lijkt, garandeert dit het volledige verstaan en onder de knie hebben van alles wat er gebeurt, een volledige onafhankelijkheid van andere code, een maximum aan flexibiliteit en een optimale performantie, zowel op computationeel als geheugengerelateerd vlak.

Met het oog op deze performantie en flexibiliteit kozen we voor de programmeertaal C++, vanwege zijn snelheid en rijke objectgeoriënteerde eigenschappen. Verder kozen we voor het GNU/Linux ontwikkelingssysteem vanwege zijn stabiliteit, veiligheid, snelheid en open source karakter. We gebruikten de `gcc-3.2.3` compiler, in combinatie met `autoconf` en `automake`. De code werd geschreven met behulp van `emacs` en bestaat uit 83 319 lijnen code (2 450 647 bytes) en 148 klassen verspreidt over 222 files.

Een aantal van de belangrijkste klassen zijn kort toegelicht in Hoofdstuk 8. Dit gaat van een random generator over Bayesiaanse en neurale netwerken tot een gedistribueerd systeem om bepaalde berekeningen op een cluster uit te rekenen. Geïnteresseerden in de code nemen best contact op met de auteur of het Departement Electrotechniek ESAT/SCD aan de Katholieke Universiteit Leuven.

Conclusies

Het centrale thema in deze thesis was hoe we verschillende soorten informatie kunnen samenbrengen in één model. Twee fundamenteel verschillende modellen werden naar voor geschoven; het ene type, Bayesiaanse netwerken, laat op een eenvoudige wijze het gebruik van achtergrondinformatie toe maar heeft beperkte leereigenschappen op basis van data. Het andere type, neurale netwerken, leert goed op basis van numerieke data maar is dan weer minder geschikt om achtergrondinformatie mee in rekening te nemen. We zouden graag de goede eigenschappen van beide modellen combineren.

Hiertoe plaatsten we beide modellen in het Bayesiaanse denkkader wat ideaal geschikt is om leerproblemen te specificeren. We ontwikkelden een methode waarbij de achtergrondinformatie omschreven wordt met behulp van een Bayesiaans netwerk. Vervolgens wordt deze informatie getransformeerd naar een *informatieve* a priori verdeling over de gewichtsruimte van een neuraal netwerk met behulp van virtuele datasets. Tenslotte wordt deze a priori verdeling geüpdatet naar de a posteriori verdeling op basis van de data.

Hierbij werd aandacht besteed aan het specificeren van de a priori informatie, het omvormen van discrete waarden naar continue en het behandelen van zowel de Bayesiaanse als de neurale netwerken in het Bayesiaanse kader. We ontwikkelden een techniek om verdelingen in de neurale netwerk gewichtsruimte te schatten, rekening houdende met de symmetrieën die hier optreden. Tenslotte werd de voorgestelde techniek toegepast op een medisch classificatieprobleem waarbij we preoperatief de kwaadaardigheid van ovariale tumoren wensen te voorspellen.

Met behulp van leercurven gebaseerd op de oppervlakte onder de ROC curve, werd het leergedrag van een uiteenlopend aantal modellen gevisualiseerd. De resultaten van de experimenten die we deden, geven duidelijk aan dat het neurale netwerk met een informatieve a priori verdeling gebaseerd op de transformatietechniek globaal gezien de beste classificatieperformantie heeft. Deze resultaten geven aan dat we op een succesvolle wijzen de informatie representatie van een Bayesiaans netwerk naar een multilayer perceptron kunnen transformeren.

We wijzen erop dat dit onderzoek op verschillende wijzen verfijnd en uitgebreid kan worden. Zo kunnen we met andere modellen de achtergrondinformatie omschrijven of leren op basis van de data. Ook het schatten van de informatieve a priori verdeling in de neurale netwerk gewichtsruimte kan verbeterd worden. Misschien bestaan er ook meer analytische methoden om de transformatie van informatie door te voeren dan onze virtuele datasetgebaseerde methode.

Op het vlak van het ovariale tumorprobleem zijn er ook nog verschillende onderzoeksthema's voorhanden. Momenteel worden andere classificatiemethoden getest waaronder support vector machines. Ook het IOTA project is verre van afgerond en nieuwe, interessante informatie is op komst. Zo werd een deel van de tumoren na de operatie bewaard en worden deze momenteel onderzocht met behulp van een micro-array. Deze technologie vormt ons een beeld van de geactiveerde genen binnen deze tumoren. Hopelijk leidt dit tot een algemeen beter inzicht in het domein van de ovariale tumoren en zet dit de deur open naar nieuwe genees- of preventiemethoden.

Contents

Voorwoord **i**

Abstract **iii**

Samenvatting **v**

Nederlandse samenvatting **vii**

Contents **xxxv**

Notation **xxxix**

Publication list **xli**

1 Introduction **1**
- 1.1 Learning from many sources of information 1
- 1.2 Deliverables and accomplishments 2
- 1.3 Chapter by chapter overview 4

2 Classification of ovarian tumours **7**
- 2.1 Ovarian tumours 8
- 2.2 Sources of information 10
  - 2.2.1 Clinical data 10
  - 2.2.2 Expert information 11
  - 2.2.3 Textual information 13
- 2.3 To conclude 14

3 The Bayesian framework **17**
- 3.1 General overview 18
  - 3.1.1 Intuitive example 19
  - 3.1.2 History 21
  - 3.1.3 Tour through the theory 24
- 3.2 Classification 28
  - 3.2.1 Three levels of decision support 29
  - 3.2.2 Making a decision 29

3.3 Performance measures . . . 33
  3.3.1 Misclassification rate . . . 34
  3.3.2 Area under the ROC curve . . . 35
3.4 To conclude . . . 36

4 Transformation of information representation 39
4.1 An example . . . 40
4.2 Donor models . . . 41
4.3 Acceptor models . . . 41
4.4 Transformation of representation . . . 42
  4.4.1 Choosing the data set size . . . 43
  4.4.2 Discrete and continuous variables . . . 44
4.5 Other approaches . . . 45
4.6 To conclude . . . 48

5 Bayesian networks 51
5.1 Description . . . 52
  5.1.1 Structure and conditional independency . . . 52
  5.1.2 Local parameters and conditional dependency models . . . 57
5.2 Algorithms . . . 58
  5.2.1 Structure learning . . . 58
  5.2.2 Parameter learning . . . 62
  5.2.3 Generating random vectors . . . 62
  5.2.4 Inference . . . 63
  5.2.5 Graph layout . . . 64
5.3 Different types of networks . . . 64
  5.3.1 Table distributed variables . . . 65
  5.3.2 Linear-Gaussian distributed variables . . . 69
  5.3.3 Nonlinear-Gaussian distributed variables . . . 71
  5.3.4 Missing values and hidden variables . . . 73
5.4 Different types of prior information available . . . 74
  5.4.1 Expert information . . . 75
  5.4.2 Textual information . . . 75
5.5 Prior information representation . . . 75
  5.5.1 Prior for the structure . . . 76
  5.5.2 Prior for the parameters . . . 77
  5.5.3 Harvesting the information sources . . . 77
5.6 To conclude . . . 79

6 Multilayer perceptrons 81
6.1 Description . . . 82
6.2 Conditional probability models and cost functions . . . 83
  6.2.1 Regression . . . 84
  6.2.2 Classification . . . 85
  6.2.3 Logistic regression . . . 86
6.3 Approximation and overtraining . . . 87

6.3.1 Overtraining 87
6.3.2 Early stopping 90
6.3.3 Weight decay and a prior distribution 90
6.4 Posterior distribution 91
6.5 Pre-processing 92
6.5.1 Continuous variables 93
6.5.2 Ordinal and nominal discrete variables 93
6.5.3 Input selection 95
6.6 Optimization 95
6.6.1 Steepest descent 96
6.6.2 Scaled conjugate gradient 97
6.7 Bayesian simulation 98
6.7.1 Weighted prior samples and rejection sampling 99
6.7.2 Markov chain methods 101
6.8 Prior for parameters 107
6.8.1 Complexity-based prior 107
6.8.2 Informative prior 109
6.9 Estimation of prior 111
6.9.1 Cause of symmetries 112
6.9.2 Symmetry elimination methods 112
6.10 To conclude 119

7 Ovarian tumour classification 125
7.1 Description 126
7.2 Sources of information 126
7.2.1 Clinical data 126
7.2.2 Expert information 128
7.2.3 Textual information 129
7.3 Learning curves and performance measure 134
7.4 Results 135
7.4.1 Previous results 136
7.4.2 Bayesian networks 137
7.4.3 Logistic regression 144
7.4.4 Multilayer perceptrons 146
7.5 To conclude 150

8 Implementation 151
8.1 General setup 151
8.2 Language and development tools 152
8.3 Main building blocks 152
8.3.1 Graphical layout of the key classes 153
8.3.2 Data 153
8.3.3 Random variables 153
8.3.4 Bayesian networks 156
8.3.5 Multilayer perceptrons 157
8.3.6 Model 157

8.3.7 Distributed computation 157
8.4 To conclude 158

9 Conclusions and perspectives 161
9.1 Conclusions 161
9.2 Future research directions 162

A Variable descriptions 165

Bibliography 175

Curriculum Vitae 181

Notation

Symbol	Meaning	Page
ϕ	Scalar	
$\boldsymbol{\phi}$	Vector	19
$\mathbf{\Phi}$	Matrix	12
$\underline{x}$	Random variable	18
$\mathrm{p}(\underline{x})$	Density of $\underline{x}$	18
$\mathrm{P}(\underline{y} = y)$	Probability that $\underline{y}$ is equal to y	18
$(\, \underline{a} \perp \underline{b} \,\vert\, \underline{c} \,)$	$\underline{a}$ is conditionally independent of $\underline{b}$, given $\underline{c}$	52
$\mathbf{D}$	a statistical data set	19
$\mathrm{E}[\underline{x}]$	Mean of $\underline{x}$	30
$\mathrm{V}[\underline{x}]$	Variance of $\underline{x}$	66
$\mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$	Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{\Sigma}$	32
$\mathcal{L}(\, \boldsymbol{\omega} \,\vert\, \mathbf{D} \,)$	Data likelihood $\mathrm{p}(\, \mathbf{D} \,\vert\, \boldsymbol{\omega} \,)$	19
$\mathcal{S}_{\mathrm{bn}}$	Bayesian network structure	58
$\boldsymbol{\theta}$	Bayesian network parameters	58
$\mathcal{S}_{\mathrm{mlp}}$	Multilayer perceptron structure	87
$\boldsymbol{\omega}$	Multilayer perceptron parameters	83
ϕ	Multilayer perceptron transfer function	82
sigmoid	The sigmoidal transfer function	87
$\mathrm{mlp}(\, \boldsymbol{x} \,\vert\, \boldsymbol{\omega} \,)$	Multilayer perceptron function	83
$\mathrm{Error}(\, \boldsymbol{\omega} \,\vert\, \mathbf{D} \,)$	The error based on the likelihood $\mathcal{L}(\, \boldsymbol{\omega} \,\vert\, \mathbf{D} \,)$	85
$\mathrm{Error}^*(\, \boldsymbol{\omega} \,\vert\, \mathbf{D} \,)$	The error based on the posterior distribution $\mathrm{p}(\, \boldsymbol{\omega} \,\vert\, \mathbf{D} \,)$	90
$\mathcal{C}_{\mathrm{P}}$	The positive classification label	29
$\mathcal{C}_{\mathrm{N}}$	The negative classification label	29
$\mathrm{g}(\, \boldsymbol{x} \,\vert\, \boldsymbol{\omega} \,)$	Binary decision function	29
$\mathrm{f}(\, \boldsymbol{x} \,\vert\, \boldsymbol{\omega} \,)$	Class probability function	29
n	Number of data vectors	29
d	Number of inputs	82
v	Number of variables	29
$\mathcal{D}$	A text document	78
$\mathcal{C}$	A set of documents	14

$\mathcal{T}(\mathcal{D})$	The vector representation of the document $\mathcal{D}$ using the term-frequency inverse-document-frequency weighting scheme	78
$\mathrm{sim}(\mathcal{D}_1, \mathcal{D}_2)$	The similarity between the documents $\mathcal{D}_1$ and $\mathcal{D}_2$	79
$\boldsymbol{V}$	A matrix containing the pairwise relations between variables	12
$\mathrm{logit}(\pi)$	The logarithm of the odds $\pi/(1-\pi)$ of the probability π	31
$\mathrm{sign}(x)$	The sign of x	74
$< \boldsymbol{x}, \boldsymbol{y} >$	The inproduct $\sum_i \boldsymbol{x}_i \boldsymbol{y}_i$ of the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$	79
$\boldsymbol{\mathcal{I}}$	The identity matrix	103
$tr(\boldsymbol{A})$	The trace $\sum_i \boldsymbol{A}_{ii}$ of the matrix $\boldsymbol{A}$	70
$\delta_a(x)$	The Dirac distribution at a	45
AUC	The area under the receiver operating characteristics curve	35

Publication list

Most of the work described in this thesis is published in one of the following publications. This work was developed in close cooperation with Peter Antal. We indicate with "⋆" those publications where Peter Antal and the author contributed equally:

- ⋆ Antal P., Fannes G., Verrelst H., De Moor B., Vandewalle J., *Incorporation of Prior Knowledge in Black-box Models : Comparison of Transformation Methods from Bayesian Network to Multilayer Perceptrons*, in Proc. of the Workshop on Fusion of Domain Knowledge with Data for Decision Support, The Sixteenth Conference on Uncertainty in Artificial Intelligence, Stanford University, USA, Jun. 2000, pp. 7-12.

- ⋆ Antal P., Fannes G., De Moor B., Vandewalle J., Van Huffel S., Timmerman D., *Bayesian predictive models for ovarian cancer classification : evalution of logistic regression, multi layer perceptron and belief network models in the Bayesian context*, in Proc. of the Tenth Belgian-Dutch conference on Machine Learning (BENELEARN2000), Tilburg, The Netherlands, Dec. 2000, pp. 125-132.

- ⋆ Antal P., Fannes G., De Smet F., De Moor B., *Ovarian cancer classification with rejection by Bayesian Belief Networks*, in Proc. of the Bayesian Models in Medicine workshop, the European Conference on Artificial Intelligence in Medicine (AIME'01), Cascais, Portugal, Jul. 2001, pp. 23-27.

- ⋆ Antal P., Fannes G., De Moor B., Vandewalle J., Moreau Y., Timmerman D., *Extended Bayesian regression models : a symbiotic application of belief networks and multilayer perceptrons for the classification of ovarian tumors*, in Proc. of the Eigth European Conference on Artificial Intelligence in Medicine (AIME'01), Cascais, Portugal, Jul. 2001, pp. 177-187.

- ⋆ Antal P., Fannes G., Timmerman D., De Moor B., Moreau Y., *Bayesian Applications of Belief Networks and Multilayer Perceptrons for Ovarian Tumor Classification with Rejection*, Journal of Artificial Intelligence in Medicine, vol. 20, no. 1-2, 2003, pp. 39-60.

- Antal P., Glenisson P., Fannes G., Mathys J., De Moor B., Moreau Y., *On the potential of domain literature for clustering and Bayesian network* *learning*, in Proceedings of the 8th ACM-SIGKDD Int. Conf. on Knowledge Discovery and Datamining, (ACM-SIGKDD), Edmonton, Canada, Jul. 2002, pp. 405-414.

⋆ Antal P., Fannes G., De Moor B., Moreau Y., *Using domain literature and data to learn Bayesian networks as clinical models of ovarian tumors*, Journal of Artificial Intelligence in Medicine, vol. 30, no. 3, 2004, pp. 257-281.

- Moreau Y., Antal P., Fannes G., De Moor B., *Probabilistic graphical models for computational biomedicine*, Journal of Methods of Information in Medicine, vol. 42, no. 2, 2003, pp. 161-168.

⋆ Antal P., Fannes G., Moreau Y., De Moor B., *Using Literature and Data to Annotate and Learn Bayesian Networks*, in Proc. of the Belgian-Dutch Artificial Intelligence Conference (BNAIC-2002), Leuven, Belgium, Oct. 2002, pp. 3-10.

Chapter 1

Introduction

1.1 Learning from many sources of information

Starting from the time when people realized they could learn, the concept itself intrigued almost everybody. Our relatively strong developed ability to learn and think is what sets us apart from other species. It is therefore not surprising that much effort went into trying to understand how learning and thinking works or how it could be simulated.

Marvellous machines have been built to accomplish the latter goal; in 1769, Baron Wolfgang von Kempelen built the “Mechanical Turk”, what von Kempelen claimed was a chess playing machine for the amusement of Queen Maria Theresa of Austria. It was said to be a purely mechanical device. Alas, it turned out to be a chess master dressed up as a wooden robot in the shape of a Turk.

As is usual in science, one tries to understand the easiest problem before proceeding with something more complicated. Understanding and implementing learning behaviour is generally thought of as being easier than thinking. The implementation of such learning behaviour has seen an explosion of interest around 1949 thanks to the work of John von Neumann, which eventually resulted into the computer we know today. Most people believe the computer is the most promising machine available today for implementing learning behaviour. The clearest example is probably the chess victory of Deep Blue over Garry Kasparov in 1997.

Vast amounts of literature are devoted to the subject of learning using a computer and numerous terms were introduced to denote it, ranging from machine learning to artificial intelligence. The problem is approached in different ways. From a probabilistic point of view, learning is regarded as the process of updating our knowledge about a certain system when new information arrives [25, 12, 86]. In a function approximation context, the emphasis is on learning a certain multi-dimensional mapping based on a set of example mappings. Other approaches try to mimic biological processes like genetic recombination (crossover and mutation) or the cellular communication occurring in the brain, both with a varying degree of realism.

Most of the available approaches today learn from information of *one* specific type. Examples are numerical representations of character images for character recognition, sound files for speech recognition, a numerical data set to perform regression, and so on. How to combine different types of information into a hybrid system or modelling heterogeneous types of prior knowledge to be used together with data is still much of an open problem, and will be the focus of this thesis.

1.2 Deliverables and accomplishments

There are lots of domains that could benefit from a technique, capable of combining the different bits and pieces of information that are present. Two major application domains we see, are the field of micro-biology, and medicine.

In the first, people try to gain insight in the working of micro-biological processes. These include the functions of a cell, the transcription mechanisms of the genes, or the functions of the proteins. Vast amounts of information have already been collected in this field. As such, the Human Genome Project mapped the complete human genome, which contain 3 billion base pairs and approximately 30 000 genes. Other techniques like micro-arrays produce similarly large chunks of information, this time concerning gene-expression levels in a certain sample. Finally, databases containing textual descriptions of genes or information related to their function, are available. Although people previously used one specific type of information, there has grown an interest for mathematical models and tools to combine different types of data.

In the medical world, huge advances have similarly been made and a lot of insight has already been gained in almost every domain. Still, medical doctors are often faced with complex problems, leading to situations where it is impossible to state a diagnosis with certainty or to specify the optimal treatment. We believe that computer-based predictive models can assist a medical expert in perfroming some of these tasks.

Further, the knowledge and information that is present, is often scattered in different formats. A medical expert acquires a lot of knowledge and experience over the years. Some of this is then written down in one of the many medical journals. And sometimes numerical data about patients is collected systematically. Unfortunately, these data sets are often expensive to collect or contain not as many patient records as we would like. One also has to be careful to ensure that the data has a good quality. As a consequence, we cannot build our models only based on the available numerical data sets but have to combine the available information.

When designing hybrid learning systems, the pragmatic side of matters is equally important as the theoretical side. Therefore, we looked for a central application where different types of information were present and different machine learning techniques could be tested upon.

We selected the medical problem where the goal is the pre-operative classification of ovarian tumours as benign or malignant tumours. This classification problem is introduced in detail in Chapter 2, together with the three main types of information that are present: statistical data, expert information (a medical doctor), and a large collection of electronic literature that describes the medical field of ovarian tumours.

Although the techniques that are developed in this thesis are not exclusively applicable to ovarian tumours, this medical field had a substantial influence on them. As such, it determined our focus on classification. Since both expert information and measurement data had to be combined, we had to design a learning approach that was capable of dealing with both. We chose for a hybrid approach, where one model is responsible for dealing with the expert and gathering its knowledge. This clearly has to be a knowledge-based model, such as a Bayesian network (see Chapter 5). The numerical data will be dealt with using a second model, which will have better learning capabilities, such as a neural network (see Chapter 6). How to combine these two models is described in Chapter 4.

The largest contribution of this work is the development of the transformation technique introduced in Chapter 4. This allows us to combine different models, and hereby combine different types of information. To realize this transformation technique, we solved the technical problem of estimating distributions in neural network weightspace and developed a technique to deal with the symmetries that occur. Other contributions consist of finding a node ordering to develop a Bayesian network, a novel technique to display a Bayesian network structure, or improvements for the nonlinear Gaussian Bayesian network model. A lot of work was invested in combining different methods and techniques to perform the necessary calculations.

On the practical side, we developed and compared different models to perform the ovarian tumour classification. Time was invested, in cooperation with Prof. Timmerman, in the development of both a structure and a parameter prior for a Bayesian network model and the preprocessing of the data.

We worked in close cooperation with Peter Antal, a Hungarian Phd student who worked at the Department of Electrical Engineering of the Katholieke Universiteit Leuven. We developed the basics of the transformation technique together and did a lot of combined research on the ovarian tumour problem, like the derivation of the expert priors. He is also the man behind the textmining part of the ovarian tumour domain.

The approach that will be introduced in this dissertation relies on different algorithms and techniques. The author invested a lot of time in the development of a software package that implements and combines the necessary techniques and is capable of performing these computations within a reasonable time frame. This resulted in an object-oriented software package capable of distributed computations for optimal performance.

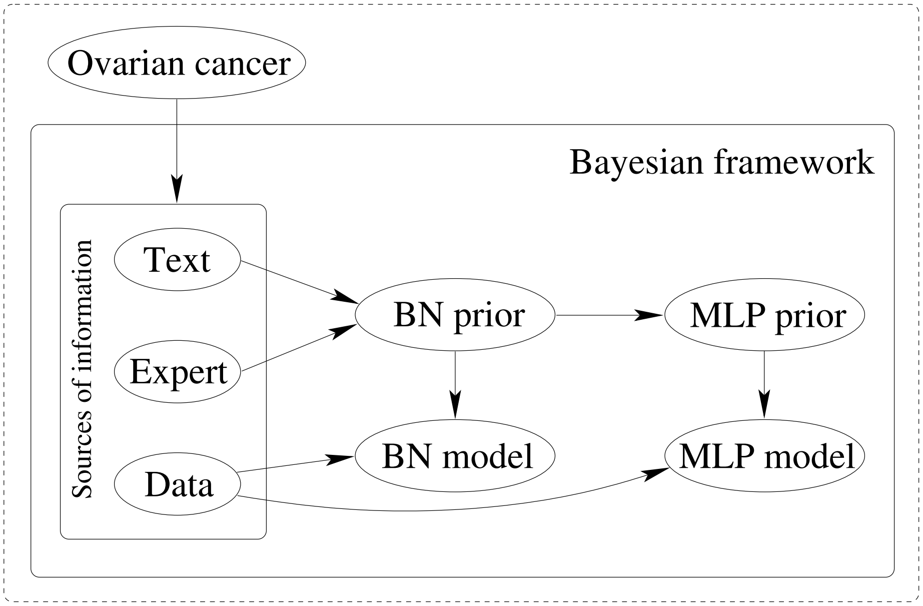


Figure 1.1: A schematic representation of the components of this thesis.

1.3 Chapter by chapter overview

A picture of the main structure of this thesis is given in Figure 1.1. It includes the major concepts and models that will be used. This figure will be repeated at the beginning of each chapter as a quick overview, while the topics discussed in that specific chapter will be indicated in bold.

Chapter 2. Classification of ovarian tumours

We start this thesis with an introduction of a medical classification problem that will serve as a test case throughout the thesis. Some general background information is presented, together with the importance of developing classification systems. Next, the variables present in this domain are introduced, together with the patient data set. The additional information that is present for this problem consists of the experience and knowledge of a medical doctor and a large collection of written medical documents from the field of gynaecological tumours.

Chapter 3. The Bayesian framework

We tried to place everything in a sound theoretical context by working in the Bayesian framework. This conceptual framework for learning is founded on probability theory and makes an extensive use of Bayes' rule, as will be described in Chapter 3. Both the history of the Bayesian framework and its difference with the frequentist approach is presented. The extended decision support offered by working in the Bayesian framework is also discussed. Since our focus is on classification, we end this chapter with the discussion of several performance measures that will be used to compare different classification systems.

Chapter 4. Transformation of information representation

This chapter is the core of the thesis, explaining how we can combine prior knowledge and statistical data. The technique relies on a *donor* model to capture the prior domain knowledge, which in our case is the expert information and the electronic documents. To obtain better learning characteristics based on statistical data, the information representation from the donor model is transformed into an informative a priori distribution for an *acceptor* model by means of virtual data sets. Technical issues like handling discrete and continuous variables are also discussed.

Chapter 5. Bayesian networks

The abstract technique introduced in Chapter 4 will be demonstrated on the ovarian tumour problem. We begin by introducing our donor model class of choice, Bayesian networks. A Bayesian network can be viewed as a graphical method to represent a joint distribution over a set of variables. Different aspects of this model are introduced: learning the structure and parameters of such a Bayesian network, generating samples from the joint distribution it represents, performing inference, and its potential for incorporating prior knowledge. This latter capability is our main motivation for selecting a Bayesian network as our donor model.

Chapter 6. Multilayer perceptrons

The counterpart of the donor model is the acceptor model. This model should have good learning capabilities from data, possibly sacrificing insight into the model parameters and thus loosing a possibility to incorporate prior knowledge. We selected multilayer perceptrons as acceptor models. This class includes logistic regression but is also capable of producing nonlinear classification boundaries. Handling these models in the Bayesian framework is discussed in detail, together with the connection between classical cost functions and the Bayesian a posteriori distribution and how we can generate parametrizations from the a posteriori distribution using the hybrid Monte Carlo Markov chain. Finally, estimating distributions in neural network parameter space and the problems that arise are discussed.

Chapter 7. Ovarian tumour classification

When the necessary models are introduced, we can apply the techniques discussed in Chapter 4 to ovarian classification. The different types of information are: the available data set, the expert information, and textual information. The construction of the informative a priori distribution for the Bayesian network model is discussed in detail, together with the data pre-processing steps necessary for the multilayer perceptron and the transformation itself between the two models.

We start the results section of this chapter with a discussion of previously applied techniques and the performance of univariate models. Next, the concept of learning curves is introduced, which serves as our main tool to visualize the learning behaviour of certain models. Using this learning curve, the learning characteristics of different models is discussed.

Chapter 8. Implementation

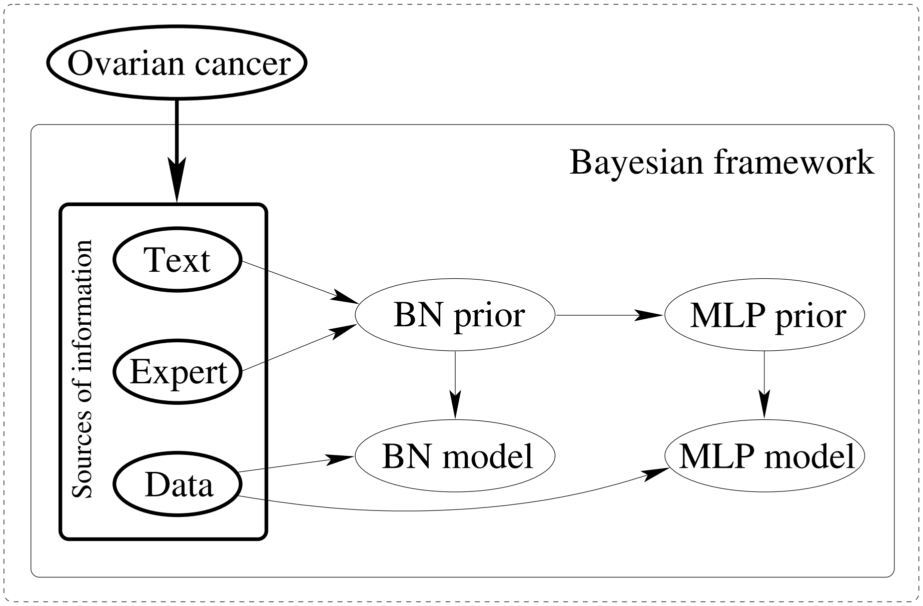
This chapter discusses the implementation details and design of the software package that was developed to perform the necessary computations.

Chapter 9. Conclusions

Finally, Chapter 9 draws some conclusions about this work, describes the accomplishments we achieved and presents an outline of a few directions that may be followed in future research.

Chapter 2

Classification of ovarian tumours



Before we dive into the theory and algorithms, we will introduce a real-world medical classification task related to cancer, together with the three main sources of information that are available. The main goal of this task consists in classifying ovarian tumours *as benign or malignant, based on certain medical observations. This task will serve as the central problem in this thesis to illustrate the developed techniques.*

2.1 Ovarian tumours

The ovaries are two small, almond-shaped organs located on either side of the uterus (see Figure 2.1), which is found in the lower part of a woman's belly. These small organs are responsible for the production of human eggs. An ovarian tumour is a growth that begins in the cells that constitute the ovaries. The three main types of cells found in an ovary are the epithelial cells or surface cells, the germ cells that are the heart of the reproductive system and produce the eggs, and the stroma cells that form the connective tissue found in the ovary and produce hormones (see Figure 2.2). Each of these cells can give rise to a specific type of tumour: epithelial tumours, which account for up to 90% of all ovarian tumours, germ cell tumours, and stromal tumours. Tumour cells that metastasize from other organ sites to the ovaries are not considered ovarian tumours. Unlike a fluid-filled cyst, an ovarian tumour is solid.

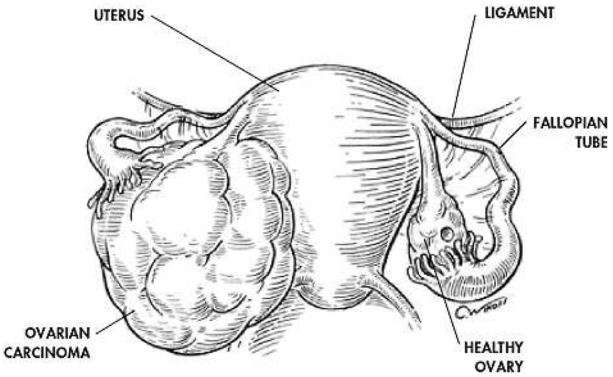


Figure 2.1: A schematic representation of the uterus and ovaries from `http://www.pdrhealth.com/`.

More important for the patient is the behaviour of the tumour. Both *benign* and *malignant* ovarian tumours exist. Benign (i.e., not cancerous) ovarian tumours occur most frequently (70% of all ovarian tumours are benign). They only rarely tend to invade tissues and never spread to other parts of the body. If untreated, however, they may grow very large and become painful. Often, these tumours will shrink with the use of birth control pills. The left part of Figure 2.3 shows a benign epithelial ovarian tumour. Malignant tumours, also called cancers, do tend to spread to other parts of the body when they grow large enough and are therefore often life-threatening. The right part of Figure 2.3 shows a malignant epithelial ovarian tumour.

According to the American Cancer Society, ovarian cancer accounts for 4 percent of all cancers among women and ranks fifth as a cause of their deaths from cancer. Unfortunately, more than two thirds of the patients are diagnosed

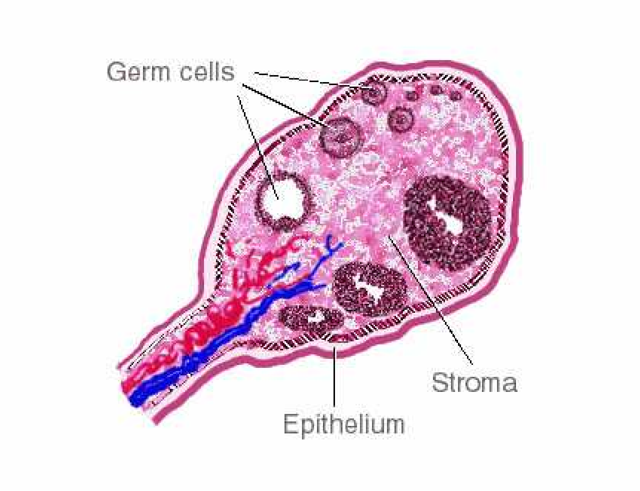


Figure 2.2: A schematic representation of an ovary and the three different types of cells from `http://www.taxol.com/`.

only at an advanced stage and therefore have poor prognosis. Therefore, early detection is of primary importance for the survival of the patient.

Ovarian tumours are fairly common. Before ultrasound techniques were routinely available, finding such a tumour was considered to be an indication for surgery. The large number of ovarian tumours now being discovered and their relatively low risk of malignancy indicates that they should not all be managed surgically. Ovarian malignancies therefore represent one of the greatest challenges among gynaecological tumours.

A reliable test to discriminate benign from malignant tumours without performing surgery would be of considerable help to clinicians to triage woman into different categories; it would help them to recognise patients for whom treatment with minimally invasive surgery or conservative management suffices versus patients who need to be referred to a gynaecological oncologist for more aggressive treatment. Developing such a test will be the main application goal of this thesis.

This research is situated in the framework of the International Ovarian Tumour Analysis Consortium (IOTA), which is a multi-centre group to study the pre-operative characterization of ovarian tumours based on artificial intelligence models [83] (`https://www.iota-group.org/`), in collaboration with the Department of Electrical Engineering at the Katholieke Universiteit Leuven (ESAT/SISTA) and various hospitals around the world. The IOTA was founded by Prof. Dr. Dirk Timmerman from the Department of Obstetrics and Gynaecology, University Hospitals Leuven in 1998. Under the name of the IOTA project, Prof. Timmerman proceeded with the prospective collection of patient data, which he already initiated in 1994. At the moment of this writing,

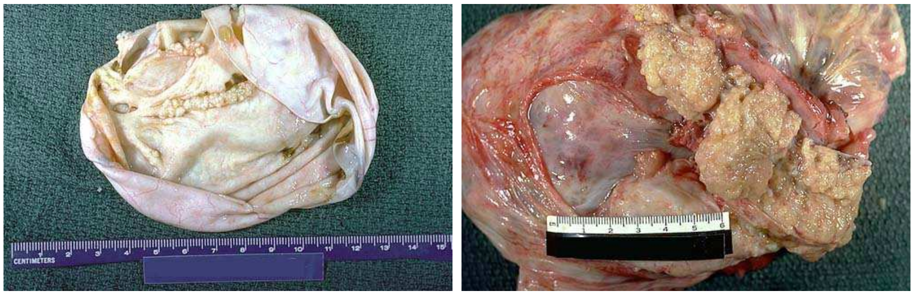


Figure 2.3: Left is a benign epithelial ovarian tumour. Right is a malignant epithelial tumour.

the database contains information from 1 346 masses and 1 152 patients which is the largest number of stage I ovarian tumours pre-operatively examined with transvaginal ultrasonography.[1]

2.2 Sources of information

There are three different types of information at our disposal to construct a predictive model for the problem described.

2.2.1 Clinical data

At first, there is a clinical data set with patient data that has been collected prospectively in the IOTA framework. This study includes the multi-centre collection of patient data and the corresponding data collection protocols. Patients with at least one adnexal mass were recruited and studied within one month before investigative surgery. The data collection protocol excludes other causes with similar symptoms, such as infection or ectopic pregnancy, and ensures that patients with a persistent extra-uterine pelvic mass undergo surgery. This procedure eliminates the possibility of false negatives. Both medical and family histories were recorded. Transvaginal ultrasonography with colour Doppler imaging was used to derive indices of tumour form and blood flow. The blood flow is characterized by a number of variables. A sample of peripheral venous blood was taken for the analyses of serum CA125. Several informative images of all the adnexal masses were made and collected digitally for quality control. Findings at surgery and the histological classification of excised tissues as malignant or benign (and by cell type) are used as outcome measures. Both techniques enhanced the quality of this study and provided us with reliable pathology values that can serve as a gold standard [83, 81].

---

[1] Although this is not the final IOTA data set, it underwent a quality control and inconsistent records were corrected by Andrea Valek.

For each tumour and patient in this data set, a total of 68 variables were collected. Based on results from previous studies [82] and with the help of our medical expert Prof. Timmerman, we selected a total of 35 variables from these 68 variables that are the most relevant ones to the domain. These include variables such as the malignancy of the tumour *(Pathology)*, parity (number of deliveries), drug treatment for infertility, use of oral contraceptives, family history of breast and ovarian tumour, age, bilaterality of the tumour, pain, descriptions of the morphology, echogenicity, vascularization of the mass, or the level of CA125 tumour marker. The pathology variable indicates if the tumour was found to be benign or malignant after surgery. For symmetric records (patients with a tumour on both left and right side) only the dominant tumour was kept. Records with missing values for one of these 35 variables were removed, except when the *Pill Use* variable was missing, for which we used an imputation model to complete the variable; we learned a linear regression model for *Pill Use* based on the samples where this variable is observed, and used this model to fill in a value for the unobserved cases.

Some variables only make sense depending on other variables. As such, the variable *CycleDay*, which represents the menstrual cycle, makes only sense for women who are pre-menopausal. In the same category, we have the *PMenoAge* and *PostMenoY* variables which only make sense for patients in the menopause. A second group deals with solid papillary projections. When these structures are present, the shape and possible flow is indicated using the variables *PapSmooth* and *PapFlow*. The last group of conditional variables deals with blood flow indices which are measured using colour Doppler imaging. This has the effect that the variables *RI*, *PI*, *PSV* and *TAMXV* are only valid when actual blood flow is present ($Colour\ Score \neq 1$).

The data set contains both continuous and discrete variables. Table 2.4 enumerates the continuous variables, their physical units, and their univariate statistics for benign and malignant tumours. For these variables, the mean and standard deviation are reported. For conditional variables, as explained in the previous paragraph, we present the statistics only for those records where the variable actually makes sense.

Table 2.5 enumerates the binary variables. For the binary variables, we computed the percentage of records that have a value of 1. In our case, depending on the variable, a 1 means *yes* or *present*, as indicated in the table.

Table 2.6 introduces the discrete variables, their possible values, and the corresponding percentages for these values that are observed in the data set. These percentages are again presented for the subgroups of benign and malignant tumours. *Colour Score* is an ordered variable, while the others are nominal.

A more in depth description for each variable can be found in Appendix A.

2.2.2 Expert information

In addition to this numerical information, we could rely on the knowledge and experience of Prof. Timmerman, a leading expert in the ultrasonography of

Variable name	Unit	Benign (70.5%)	Malignant (29.5%)
Age	year	46.07 (15.26)	55.54 (14.59)
CA125	U/ml	37.84 (91.97)	840.9 (2493)
Fluid	mm	2.527 (6.825)	15.4 (19.54)
PI (Colour Score $\neq 1$)		1.101 (0.656)	0.819 (0.387)
PSV (Colour Score $\neq 1$)	cm/sec	15.36 (15.75)	31.26 (22.53)
Parity		1.223 (1.322)	1.613 (1.387)
Pill Use	year	3.234 (4.828)	2.822 (4.888)
RI (Colour Score $\neq 1$)		0.604 (0.141)	0.517 (0.153)
Septum	mm	1.336 (2.148)	2.842 (3.852)
TAMXV (Colour Score $\neq 1$)	cm/sec	9.502 (11.11)	21.55 (16.38)
Volume	ml	302.9 (777.5)	683 (1160)

Figure 2.4: The univariate statistics of the IOTA data set for the continuous variables. For these variables, the sample mean is presented, together with the standard deviation and their physical units. These statistics are presented for the subgroup of benign tumours (the left column) and the subgroup of malignant tumours (the right column).

ovarian tumours and the founder of the IOTA project. Working at the University Hospitals Leuven, he has collected the main part of the IOTA data set.

He helped selecting the relevant variables for our experiments and their corresponding discretization bins. In addition to this, he was able to construct a model describing the relations between the medical variables in the form of a Bayesian network structure for a subset of 11 variables and presented us with the parametrization for this network structure by answering a questionary consisting of 29 pages, as described in Section 5.5.3. Figure 2.7 displays this expert network structure.

Besides this parametrized prior Bayesian network, he was also able to specify the pairwise relationships between all the variables, which was transformed to a prior distribution over the network structure space using the technique explained in Section 5.5.3. This results in a matrix $\boldsymbol{V}$ where element $\boldsymbol{V}_{ij}$ is an indication of the strength of the relation between the variables $\underline{x}_i$ and $\underline{x}_j$. In Section 5.5.1, we will use this as the a priori probability to see an edge from variable $\underline{x}_j$ to variable $\underline{x}_i$ in a Bayesian network. This matrix is displayed in Figure 2.8, where the grey-scale and the thickness of an edge between $\underline{x}_i$ and $\underline{x}_j$ corresponds to the value $\boldsymbol{V}_{ij}$ from the matrix.

Finally, our expert provided us with keywords and full text descriptions of the domain variables and a selection of relevant journals for the domain. Using these directions, we could use the electronic literature as an additional information source.

Variable name	Benign (70.5%)	Malignant (29.5%)
Ascites = present	3.3%	45.2%
Bilateral = yes	15.2%	31.7%
FamHistOCCa = yes	2.2%	10.3%
FamHistBrCa = yes	5.6%	13.5%
HormTherapy = yes	24.5%	17.8%
IncomplSeptum = yes	10.1%	3.9%
Pain = present	26.6%	17.8%
PapFlow (Pap = yes) = yes	27.8%	84.8%
PapSmooth (Pap = yes) = yes	48.1%	86.7%
Papillation = yes	19.6%	45.7%
PersHistBrCa = yes	2.4%	5.7%
Shadows = present	11.1%	1.3%
Solid = yes	41.1%	93.5%

Figure 2.5: The univariate statistics of the IOTA data set for the binary variables. For these variables, the percentage of records with a value of 1 is given. This 1 means *yes* or *present*. These statistics are presented for the subgroup of benign tumours (the left column) and the subgroup of malignant tumours (the right column).

2.2.3 Textual information

A last and quite extensive source of information is contained in textual documents that report about the domain. With *domain*, we mean the specific application field we are considering. In this thesis, this is the field of ovarian tumours.

We will use this literature information to discover relationships between domain variables. The rationale in the medical context is that a significant body of medical research is devoted to the discovery of informational relationships between domain variables. Mainly dependencies are reported in the literature and independencies are largely ignored. The usage of text-mining methods for relationship extraction has already proved useful in providing insight for the domain expert and data analyst in many domains [80, 14, 50, 49]. We will take this one step further and use the results of these methods automatically in the statistical learning of quantitative models.

Instead of following a linguistic approach, which ambitiously tries to structurally analyse [67] and extract high level logical statements from free text [73, 70, 21], unstructured statistical approaches have similarly shown good performance in extracting entity relationships.

The whole corpus of available documents can be divided into two main groups. The first group contains documents that describe one specific domain variable. Included are the name of the variable, synonyms, keywords, the IOTA protocols for that variable, medical dictionary definitions (Merck Manual, CancerNet and On-line Medical dictionary), the definitions from our medical expert [81], and references to other electronic literature.

Variable name	Benign (70.5%)	Malignant (29.5%)
Colour Score		
None	39.6%	6.5%
Minimal	27.3%	12.6%
Moderate	27.1%	41.7%
Very strong	6.0%	39.2%
Locularity		
Unilocular	36.3%	0.4%
Unilocular solid	12.1%	16.6%
Multilocular	22.7%	6.1%
Multilocular solid	21.7%	45.2%
Solid	7.2%	31.7%
Meno		
Pre	60.9%	33.5%
Hysterectomy	7.6%	11.3%
Post	31.5%	55.2%
WallRegularity		
Regular	64.1%	16.5%
Irregular	33.2%	83.1%
Wall not visible	2.7%	0.4%

Figure 2.6: The univariate statistics of the IOTA data set for the discrete variables. The different values each variable can take are indicated, together with the percentages of that specific value for both subclasses of benign and malignant tumours.

The second group consists of documents that are not bound to one specific domain variable, but describe the relation between several variables. This group comprises entries from the MEDLINE collection of abstracts of the US National Library of Medicine. We asked our medical expert to select three sets of journals, ranging from very relevant to less relevant. The corresponding document corpora are denoted with $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$, and contain respectively 5 367, 71 845 and 378 082 abstracts dated between January 1982 and November 2000.

2.3 To conclude

In this chapter, we briefly discussed the ovarian tumour domain in general and motivated the importance of a pre-operative classification of these tumours. We pointed out three different types of information we plan to integrate, namely the statistical data set collected in the IOTA framework, the knowledge and experience of the medical expert Prof. Timmerman, and a large set of electronic documents that discuss part of the domain.

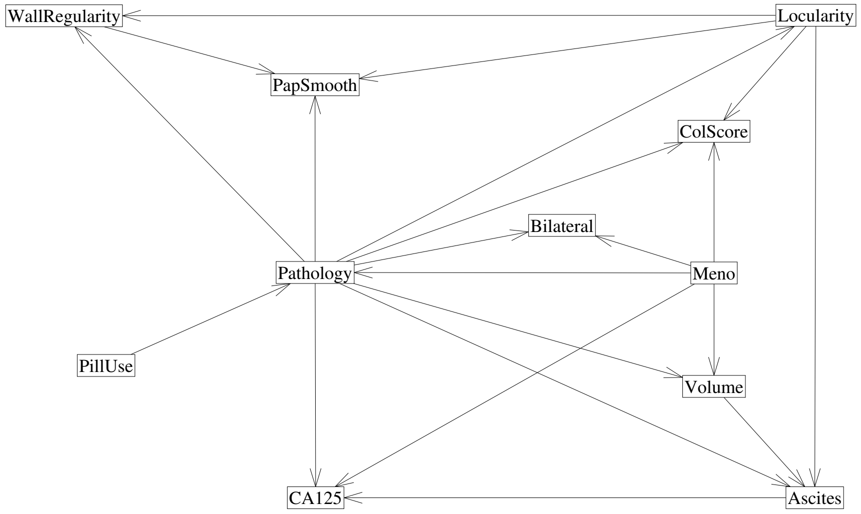


Figure 2.7: The network structure for the ovarian tumour problem containing 11 variables, constructed by Prof. Timmerman.

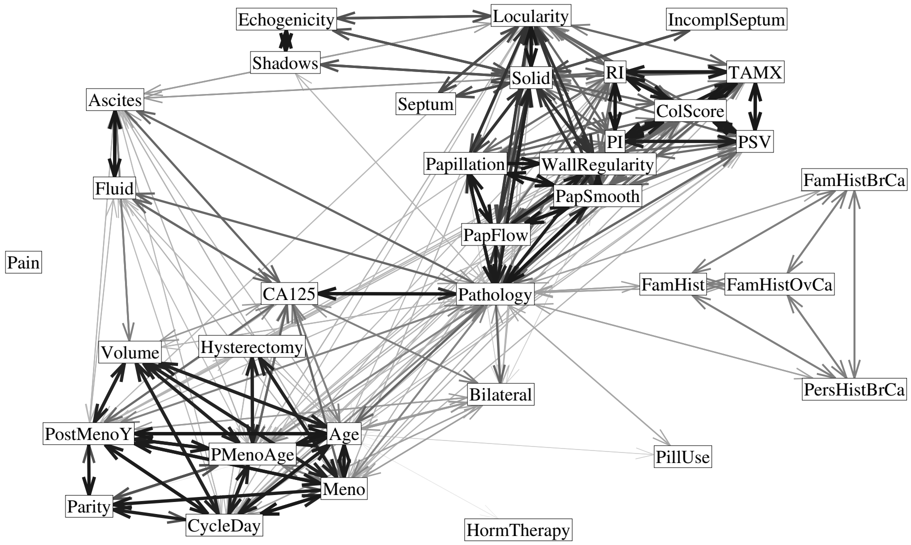
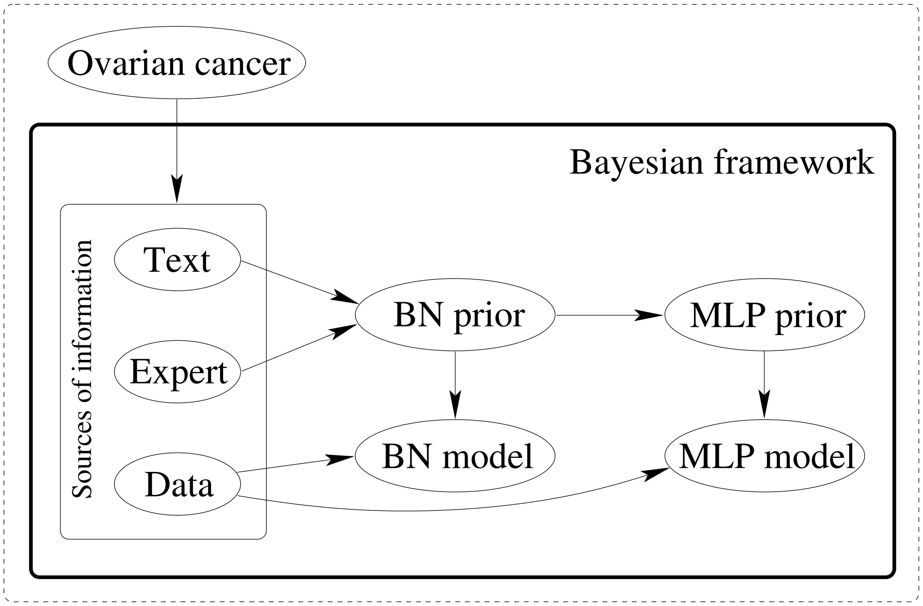


Figure 2.8: A representation of the matrix $\boldsymbol{V}$ where element $\boldsymbol{V}_{ij}$ is an indication of the strength of the relation between $\underline{x}_i$ and $\underline{x}_j$, constructed by Prof. Timmerman.

Chapter 3

The Bayesian framework



When knowledge has to be extracted from a certain set of observations, we are faced with a learning task, e.g. the central problem of classifying ovarian tumours that was introduced in the previous chapter. We will deal with this in Bayesian framework*, which is a conceptual framework for learning derived from Bayes' rule of probability. We are indeed allowed to apply probability theory to perform inference and plausible reasoning instead of restricting its use to calculating frequencies of random variables [20, 47]. This means that we can use probabilities to describe our current knowledge, and use the rules of probability theory to update this knowledge when additional information is acquired.*

3.1 General overview

In the Bayesian framework, a probability is a *degree of belief* that some statement is true. This belief *always* depends on the given knowledge ξ and can even be applied on quantities that are not intrinsically random; some things can be predicted with certainty, but when the necessary information to do this is missing, we have to resort to plausible reasoning. In a game of darts, it is possible to compute if someone will hit the bull's eye, but only if the initial speed and location vectors are known.

Richard Cox wondered what kind of systems could be used for learning and reasoning and started from two basic axioms, a consistency requirement and the implicit assumption to represent probabilities with real numbers. His first axiom states that if we specify our belief that some statement is true, we must have stated implicitly our belief that the statement is false. If you know the probability that someone will hit the bull's eye, you immediately know the probability that he or she will *not* hit the bull's eye.

The second axiom says that if we state our belief that a first statement is true, together with our belief that a second statement is true given that the first is true, that we must have stated implicitly our belief that both statements are true. If you know the probability that someone will hit the bull's eye together with the probability that this person will win the game if he hits the bull's eye, you should know the probability that he will both hit the bull's eye and win the game.

Finally, the consistency constraint ensures that if we have different ways to use the same information, we finally should end up with the same result.

As Richard Cox proved in 1946 [20], each system that is conform with these axioms can always be transformed to the system of probability theory with the basic *sum* and *product* rule:

$$\sum_a \mathrm{P}(\,\underline{a} = a \,|\, \xi\,) = 1 \quad \text{with} \quad \mathrm{P}(\,\underline{a} = a \,|\, \xi\,) \geq 0$$

$$\mathrm{P}(\,\underline{a} = a, \underline{b} = b \,|\, \xi\,) = \mathrm{P}(\,\underline{a} = a \,|\, \underline{b} = b, \xi\,)\,\mathrm{P}(\,\underline{b} = b \,|\, \xi\,).$$

The summation in the sum rule goes over a mutually exclusive and exhaustive set of values for the random variable $\underline{a}$. We denote with $\underline{a}$ a random variable, which can be a logical proposition or a set function defined on a σ-field in the Kolmogorov spirit [56]. We denote with ξ the background knowledge that is present, the set of all information given when the statements are made.

We stated the sum and product rules in a discrete setting. When dealing with continuous variables, we are working with probability density functions and have to exchange the summation with an integral. Probability density functions will be denoted with the symbol p, while discrete probabilities are denoted with the symbol P.

Using these basic sum and product rules, a lot of useful results can be derived.

The most useful equations for this thesis are

$$\mathrm{P}(\,\underline{a} = a \,|\, \underline{b} = b\,) = \frac{\mathrm{P}(\,\underline{b} = b \,|\, \underline{a} = a\,)\,\mathrm{P}(\underline{a} = a)}{\mathrm{P}(\underline{b} = b)} \tag{3.1}$$

$$\begin{aligned} \mathrm{p}(\,\boldsymbol{\omega} \,|\, \mathbf{D}, \xi\,) &= \frac{\mathrm{p}(\,\mathbf{D} \,|\, \boldsymbol{\omega}, \xi\,)\,\mathrm{p}(\,\boldsymbol{\omega} \,|\, \xi\,)}{\mathrm{p}(\,\mathbf{D} \,|\, \xi\,)} \qquad (3.2) \\ &\propto \mathrm{p}(\,\mathbf{D} \,|\, \boldsymbol{\omega}\,)\,\mathrm{p}(\boldsymbol{\omega}) \end{aligned}$$

and

$$\mathrm{P}(\,\underline{a} = a \,|\, \xi\,) = \sum_b \mathrm{P}(\,\underline{a} = a, \underline{b} = b \,|\, \xi\,). \tag{3.3}$$

The first equality — Equation 3.1 — is due to Thomas Bayes [8] and is the core of the Bayesian framework. Equation 3.2 states exactly the same as Equation 3.1, but with meaningful symbols in place; we assume some probability distribution $\mathrm{p}(\,\cdot \,|\, \boldsymbol{\omega}\,)$ with parameter vector $\boldsymbol{\omega}$ that describes the distribution of the individual data records, a data set $\mathbf{D}$, and background knowledge ξ.

Often, we are interested in how our belief (the probability distribution) in the model parameters $\boldsymbol{\omega}$ changes when a certain data set is observed. Before we observe this data set, our model parameter distribution is given by the *a priori distribution* $\mathrm{p}(\,\boldsymbol{\omega} \,|\, \xi\,)$. Bayes' rule now states that we should update this prior distribution to the *a posteriori distribution* by multiplying with the data likelihood $\mathrm{p}(\,\mathbf{D} \,|\, \boldsymbol{\omega}, \xi\,) = \mathcal{L}(\,\boldsymbol{\omega} \,|\, \mathbf{D}\,)$ and normalizing with $\mathrm{p}(\,\mathbf{D} \,|\, \xi\,)$. This procedure is depicted schematically in Figure 3.1. Almost always we can drop the ξ information from the likelihood since the model parameters are the only information we need to calculate the data distribution. The normalization constant can also often be dropped, since it is independent of the model parameters $\boldsymbol{\omega}$. We indicate with the symbol $\propto$ whenever we neglect a constant multiplicative factor.

The second formula — Equation 3.3 — is called *marginalization* and will be often used to introduce new variables or cancel variables out. The summation is over a mutually exclusive and exhaustive set of possibilities. When the random variable $\underline{b}$ is continuous, it will have a probability density and we have to replace the summation with an integral.

3.1.1 Intuitive example

To demonstrate Bayes' rule, we start with a easy medical example. Suppose there exists a certain disease in combination with a test to check if some person has the disease. The fact that someone has the disease or not is indicated with the binary random variable $\underline{d}$, where we denote with $\underline{d} = 1$ that someone has the disease. The outcome of the test will similarly be represented with a random variable, called $\underline{t}$ this time. Similarly, $\underline{t} = 1$ corresponds to the event that the test indicates that a person has the disease (the test can be wrong of course). Suppose that one out of one thousand people has the disease and that the test makes a mistake one time out of one hundred, no matter if the person has the disease or not. In symbols, this means that $\mathrm{P}(\underline{d} = 1) = 1/1000$

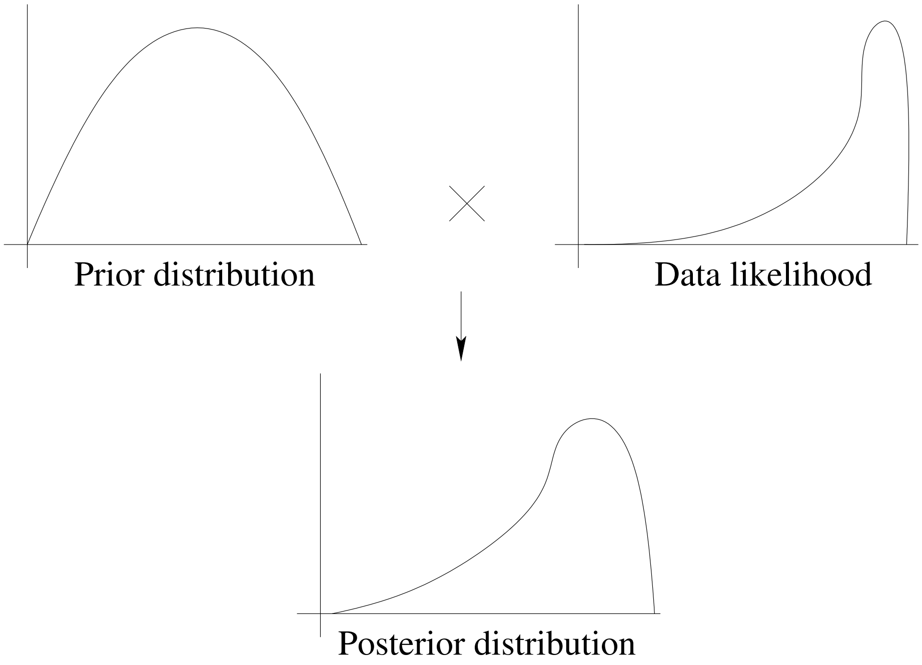


Figure 3.1: From a prior distribution to a posterior distribution: we have to multiply the prior distribution with the data likelihood to obtain the posterior distribution, up to the normalization factor $\mathrm{p}(\,\mathbf{D}\,|\,\xi\,)$.

and $\mathrm{P}(\,\underline{t} = 1\,|\,\underline{d} = 1\,) = \mathrm{P}(\,\underline{t} = 0\,|\,\underline{d} = 0\,) = 99/100$. Next, we apply the test to a random person and find out that the test is positive ($\underline{t} = 1$). Should our unfortunate person start to despair?

To answer this question, we are interested in the probability $\mathrm{P}(\,\underline{d} = 1\,|\,\underline{t} = 1\,)$ that the person has the disease, given that the test resulted positive. We can compute this probability by applying Bayes' rule and marginalization:

$$\begin{aligned}
\mathrm{P}(\,\underline{d}=1\,|\,\underline{t}=1\,) &= \frac{\mathrm{P}(\,\underline{t}=1\,|\,\underline{d}=1\,)\,\mathrm{P}(\underline{d}=1)}{\mathrm{P}(\underline{t}=1)} \\
&= \frac{\mathrm{P}(\,\underline{t}=1\,|\,\underline{d}=1\,)\,\mathrm{P}(\underline{d}=1)}{\mathrm{P}(\,\underline{t}=1\,|\,\underline{d}=0\,)\,\mathrm{P}(\underline{d}=0)+\mathrm{P}(\,\underline{t}=1\,|\,\underline{d}=1\,)\,\mathrm{P}(\underline{d}=1)} \\
&= \frac{\frac{99}{100}\frac{1}{1000}}{\frac{1}{100}\frac{999}{1000}+\frac{99}{100}\frac{1}{1000}} \\
&\approx 0.0902 = 9.02\%.
\end{aligned}$$

Although the test is fairly accurate, it is not able to update our prior belief of 0.1% that a random person has the disease ($\mathrm{P}(\underline{d} = 1) = 1/1000$) much higher than 9%. Our prior belief in the disease is simply too small. If this prior belief was higher, such that one out of only one hundred persons had the disease

($P(\underline{d} = 1) = 1/100$), our posterior belief in the disease after a positive test would rise to 50%. Adjusting the prior belief could for instance be done by looking only at a special subset of persons with a higher risk for the disease.

This example demonstrates one of the possible uses of Bayes' rule and indicates how prior knowledge should put observations in the right context. The prior knowledge in this example is the probability $P(\underline{d} = 1)$ that a random person has the disease, while the observation is the outcome of the test. Even though the test is fairly accurate, a random person should see its outcome in the right perspective, namely that the disease is very uncommon. If a person with an already elevated risk of having the disease, observes the same test result, things change drastically.

3.1.2 History

The Bayesian view on probabilities was subject to a lot of debate and definitely has had a turbulent past. We will briefly summarise its history before we continue with more elaborate applications and properties.

When serious reasoning with uncertainty originated some 300 year ago, Jacob Bernoulli (1654–1705) was among the first to wonder how *deductive logic* could be used to perform *inductive logic*. Deductive logic is that type of reasoning that deduces the possible outcomes or observations starting from a certain cause. Most reasoning done in the field of pure mathematics, where one starts from a few axioms (the causes) and uses these axioms to deduce and prove certain theorems (the observations), is an example of deductive logic where no uncertainty is involved. Most chance games serve as another example, with uncertainty involved this time. Starting from well defined — but hard to find — objects like fair coins and dice, independent repetitions and the absence of relevant prior information (the causes), we can assign probabilities to various statements (the observations).

Inductive logic on the other hand, tries to accomplish the inverse of deductive logic by predicting the possible causes when certain observations are made. Most real-world problems and every day challenges are of this type, and less straightforward to solve. Both types of logic are depicted in Figure 3.2. The top diagram illustrates deductive logic where one reasons from cause to observation. The bottom diagram illustrates the nature of inductive logic, where we are interested in deriving information about the possible causes that could have given rise to the observations.

It was reverend Thomas Bayes (1702–1762) who provided the answer to Bernoulli's problem, although only after his death through a paper sent to the Royal Society by Richard Price, a friend of Bayes', and published in 1763 [8]. These results where accepted by Pierre-Simon Laplace (1749-1827) in a 1781 memoir, and even rediscovered by Condorcet. Laplace presented them in greater clarity than Thomas Bayes did and in the form of Equation 3.1 in his book *Théorie Analytique des Probabilités*, together with applications in celestial mechanics, life expectancy and the length of marriages, and even jurisdiction [57].

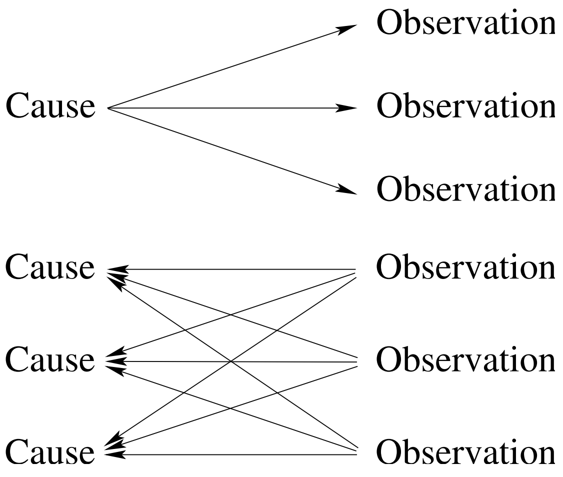


Figure 3.2: The diagram on the top illustrates deductive logic, while the bottom illustrates the nature of inductive logic.

For Bayes and Laplace, a probability represented a *degree of belief* or *plausibility* that a certain statement is true. Despite the practical successes of Laplace with the theory, their definition of the concept of probability was found too vague and too subjective for most scientists to accept since the belief of one person could be different from that of another. Therefore, a new definition of the concept of probability was introduced by John Venn (1834–1923) [87] in 1866 as the long-run relative frequency of a certain event in an effort to make the concept more objective. Note that the problem lies within the concept of probability and not with Bayes' rule of probability.

This new definition indeed feels more objective, but it can be very superficial to apply and limits the usage of probability theory. To point this out, we briefly discuss one of the practical applications discussed by Laplace, namely how to estimate the mass of Saturn, based on orbital data. Laplace realized that he was basically interested in the posterior density of this mass given the observed data together with the knowledge of classical mechanics. With the help of Bayes' rule, he was able to compute this probability density function. Figure 3.3 gives a schematic representation of this probability density that Laplace obtained. Using this density, he was able to compute the probability that the mass was between a and b, by integrating over this interval.

If the mass of our planet is viewed as a constant instead of a random variable, we cannot use probability theory in the frequentist framework though since there *is* no frequency distribution for this mass. We can introduce a frequency distribution, but then we have to interpret Figure 3.3 as the empirical distribution of the mass for many repetitions of our universe where the same orbital observations are made.

As the latter is quite unpractical to work with, the route of statistics is

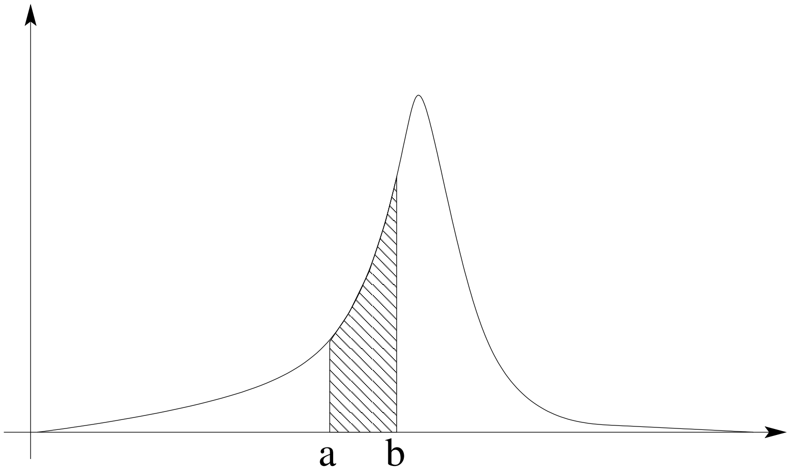


Figure 3.3: A schematic representation of the posterior probability density of the mass of Saturn that Laplace was able to compute using orbital data and knowledge about classical mechanics. The probability that the mass lies between a and b can be obtained by integrating the density over the interval $[a, b]$

followed: the mass of the planet is related to the orbital data in some way through a *statistic*, a function of the observed data. Since the data *is* subject to random noise, our statistic will become a random variable to which frequentist probability theory can be applied. Unfortunately, clear and natural rules to choose this statistic are lacking. One of the most prominent defenders of the frequentist framework was sir Ronald Fisher.

Due to the work of sir Harold Jeffreys (1891–1989) [48], it seems more natural again that randomness represents our lack of knowledge about a system, which results in our inability to predict with certainty. A nice example is a pseudo-random number generator. The internal working of such a generator is completely deterministic, but without the knowledge of the algorithm and the seeds we cannot predict it.

This lack of knowledge concept is conform to the plausibility definition of a probability proposed by Bayes and Laplace. The subjectivity is more a confusion between subjectivity and the difficult question of how probabilities should be assigned. Objectivity demands only that two people who have the same information at their disposal should end up with the same probabilities [47].

Finally, in 1946, Richard Cox (1898–1991) started from plausible reasoning and a few general assumptions and could prove that probability theory is the *only* set of rules that can be used for this [20]. This extends the usability of probability theory from frequencies to the basic calculus for logical and consistent plausible reasoning. Edwins Jaynes (1922–1998) built on these results in his last book "Probability theory, the logic of science" [47].

3.1.3 Tour through the theory

Let us assume we have a possibly biased die $\underline{d}$ and want to model this object in the Bayesian framework. We start with choosing the model class we will restrict us to. This model class is a family of distributions that we consider to be possible. Often, this model class corresponds to a set of distributions of a certain mathematical type, like the set of Gaussian distributions or the set of exponential distributions etc. We assume that we can parametrize our model class; each element of the model class can then be represented with a certain parametrization $\boldsymbol{\omega}$. Working in the Bayesian framework means that we create a model from our model class by specifying our belief in each individual parametrization. A model in the Bayesian framework is thus a set of distributions (the model class) and a distribution $\mathrm{p}(\boldsymbol{\omega} \,|\, \xi)$ specifying our degree of belief for each element of the model class.

We assume that our die is time independent and can therefore represent it with a table distribution with six possible outcomes. Such a table distribution is a discrete distribution over a finite set of possible outcomes. The model contains the probabilities for each different possible outcome. The model class is thus the set of all possible table distributions, while the parameters of this table distribution are the probabilities of the different outcomes of the die

$$\mathrm{p}(\underline{d} = i \,|\, \boldsymbol{\omega}) = \boldsymbol{\omega}_i \text{ with } i \in \{1, \ldots, 6\} \text{ and } \sum_i \boldsymbol{\omega}_i = 1$$

where $\boldsymbol{\omega}_i$ simply represents the probability to throw an i.

How should we select our prior distribution for these different probabilities? This distribution should represent all our knowledge about the die before we start using it. This depends on the prior knowledge we have and could for instance depend on the fact that the die comes from a louche casino or a game of monopoly, whether one side is more worn than another, etc. If no information at all is available, we could select a uniform distribution for the probabilities, subject to the fact that they should sum up to unity.

A common choice for this prior distribution is the Dirichlet distribution (see Section 5.3.1 for more information on this distribution)

$$\mathrm{p}(\boldsymbol{\omega} \,|\, \xi) = \begin{cases} \mathrm{C}^{\underline{\mathrm{st}}} \prod_i \boldsymbol{\omega}_i^{m_i - 1} & \sum_i \boldsymbol{\omega}_i = 1, \ \ \boldsymbol{\omega}_i \geq 0 \\ 0 & \text{else.} \end{cases}$$

Note that the density is non-zero only for $\boldsymbol{\omega}$ satisfying the constraints. The constant $\mathrm{C}^{\underline{\mathrm{st}}}$ is set up to ensure that the integral of the density over the parameter space is equal to one:

$$\mathrm{C}^{\underline{\mathrm{st}}} = \frac{\Gamma(\sum_i m_i)}{\prod_i \Gamma(m_i)} = 1 / \iint_{\sum_i \boldsymbol{\omega}_i = 1} \prod_i \boldsymbol{\omega}_i^{m_i - 1} d\boldsymbol{\omega}_i .$$

We call the parameters m_i the prior counts. Figure 3.4 shows the marginal distribution of a two-dimensional Dirichlet with different prior count settings.

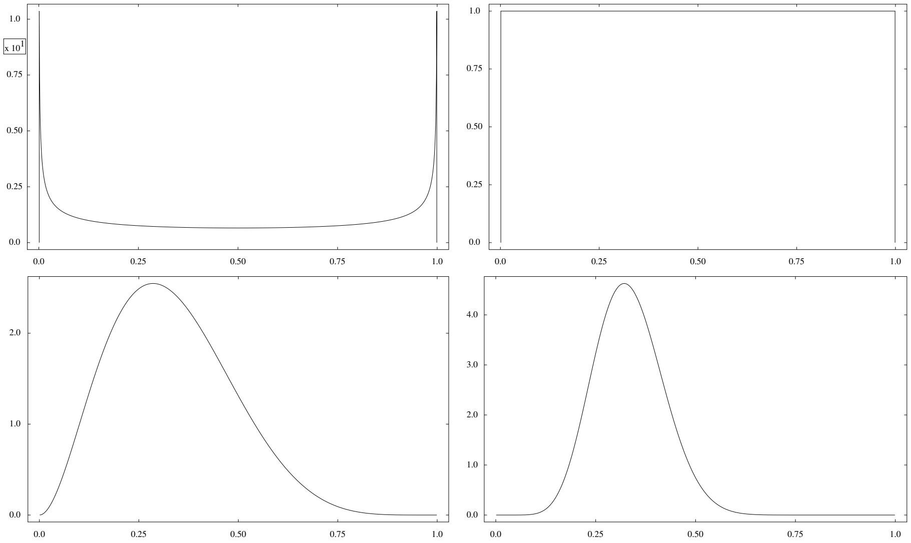


Figure 3.4: The marginal distribution of the first element of a two-dimensional Dirichlet random vector for different prior count settings. The top left has prior count settings (0.5,0.5), while the top right has settings (1.0,1.0). The bottom figures correspond to prior count settings (3.0,6.0) and (10.0,20.0).

If we set all the prior counts to 1.0, we end up with the uniform distribution over the space of positive numbers that sum up to 1. It will be explained in Section 5.5.3 how these prior counts have to be chosen to express other prior beliefs in the parameters.

We can now express our posterior belief in the model parameters when a certain data set is observed

$$\begin{aligned} p(\boldsymbol{\omega} \mid \mathbf{D}, \xi) &= \frac{p(\mathbf{D} \mid \boldsymbol{\omega})\, p(\boldsymbol{\omega} \mid \xi)}{p(\mathbf{D} \mid \xi)} \\ &\propto \prod_j p(\underline{d} = d_j \mid \boldsymbol{\omega})\, p(\boldsymbol{\omega} \mid \xi) \\ &= \prod_i \boldsymbol{\omega}_i^{n_i} \prod_i \boldsymbol{\omega}_i^{m_i - 1} \\ &= \prod_i \boldsymbol{\omega}_i^{n_i + m_i - 1} \\ &= \prod_i \boldsymbol{\omega}_i^{\#_i - 1}. \end{aligned}$$

Note that the product over j iterates over the data elements, namely the actual die rolls d_j, while i goes over the six possible outcomes of the die. The number of times the we see a die roll with outcome $\underline{d} = i$ in our data set is indicated with n_i: $n_i = |\{j \mid d_j = i\}|$. We denote with $\#_i$ the sum of the data count n_i and the prior count m_i, which can be interpreted as the total number of times we saw a die roll with result i, either as prior knowledge or during the experiments. As a final remark, we observe that the likelihood of the data has the form of a multinomial distribution.

This posterior distribution over the model parameters tells *everything* about our updated belief in the different possible parametrizations of our die, and can be used to compute certain things of interest, like the probability that the next throw will be one:

$$\begin{aligned}
\mathrm{P}(\,\underline{d} = 1 \,|\, \mathbf{D}, \xi\,) &= \int \mathrm{P}(\,\underline{d} = 1 \,|\, \mathbf{D}, \xi, \boldsymbol{\omega}\,)\, \mathrm{p}(\,\boldsymbol{\omega} \,|\, \mathbf{D}, \xi\,)\, d\boldsymbol{\omega} \\
&= \frac{\Gamma(\sum_i \#_i)}{\prod_i \Gamma(\#_i)} \int \boldsymbol{\omega}_1^{\#_1} \prod_{i=2}^{6} \boldsymbol{\omega}_i^{\#_i - 1} d\boldsymbol{\omega} \\
&= \frac{\Gamma(\sum_i \#_i)}{\prod_i \Gamma(\#_i)} \frac{\Gamma(\#_1 + 1) \prod_{i=2}^{6} \Gamma(\#_i)}{\Gamma(\sum_i \#_i + 1)} \\
&= \frac{\#_1}{\sum_i \#_i}. \qquad (3.4)
\end{aligned}$$

If we assume absence of any prior knowledge in the previous example, we have to set the prior counts m_i all to one. This results in the following formula for the probability to throw a one:

$$\mathrm{P}(\,\underline{d} = 1 \,|\, \mathbf{D}, \xi\,) = \frac{n_1 + 1}{\sum_i n_i + 6}.$$

The following example performs these computations for two different dice. The first die, $\underline{d}_1$, is generated from a uniform Dirichlet distribution (all pseudo-counts are set equal to 1.0). The second die, $\underline{d}_2$, is generated from a Dirichlet distribution with all pseudo-counts equal to 50.0. This will, in general, result in a die that is more honest than $\underline{d}_1$. The two generated dice are characterized by their probabilities for each of the six possible outcomes. These probabilities can be read from the following table:

Outcome	1	2	3	4	5	6
$\mathrm{p}(\underline{d}_1)$	0.0736	0.2029	0.4513	0.1892	0.0156	0.0671
$\mathrm{p}(\underline{d}_2)$	0.1376	0.1754	0.1892	0.1630	0.1887	0.1459

Next, we threw 100 times with each dice, storing the outcomes of die $\underline{d}_i$ in the data set $\mathbf{D}_i$. The outcomes of these experiments are given here under:

	Outcomes
D$_1$	3, 2, 3, 3, 3, 4, **1**, 4, 3, 2, 3, 3, 3, 3, 2, 3, 4, 2, 3, 2, **1**, 2, 2, 3, **1**,
	3, 2, 3, 2, 3, 3, 3, 2, 4, 4, 3, 3, 2, **1**, 4, 2, 3, 4, 3, 5, 4, 2, 2, 5, 3,
	2, 2, 4, 3, 3, 3, 2, 4, 3, 3, 2, 4, **1**, 2, 5, 3, 3, 3, 3, 4, 4, 3, 4, 4, 3,
	3, 3, 3, 6, 3, 4, 3, 2, 3, 3, 3, 3, 3, 3, 3, 2, 3, 2, 3, 3, 3, 4, 4, 3, 3
D$_2$	3, 4, **1**, 4, 4, 4, 4, 5, 3, **1**, 3, 6, 5, 2, 4, 2, 3, 3, 6, 3, 6, **1**, 6, 5, 3,
	5, 2, 2, 3, **1**, **1**, 3, **1**, 5, 5, 2, **1**, 5, 2, 2, 3, 6, 5, 5, 4, 6, 4, 2, **1**, 3,
	6, 5, **1**, 5, 5, 5, 5, 2, 2, 3, 4, 4, 5, 6, 3, **1**, 2, **1**, 5, **1**, 4, 5, 3, 2, 2,
	2, 6, **1**, 4, 5, 4, 4, 5, 5, 2, 3, 5, **1**, 6, 4, 5, 5, 3, 3, 5, **1**, **1**, **1**, 6, 6

Finally, we estimate for each data set the probability that the next throw will be a one. The data counts for observing a one (n_1) are five for the first die and 17 for the second die (indicated in bold). The prior counts depend on which prior we use. The prior counts m_i for the uniform prior are all 1.0, while those for the honest prior are all equal to 50.0. We compute the probability to throw a one with Equation 3.4 using both the uniform prior and the honest prior for each data set. Remember that $\#_i$ was the sum of the prior count m_i and the data count n_i.

	Uniform prior	Honest prior	True probability
D$_1$	**0.0566**	0.1375	0.0736
D$_2$	0.1698	**0.1675**	0.1376

We see that using the prior that was used to generate that actual die, gives the best probability estimate. The value that was closest to the true probability is indicated in bold. The uniform prior is very flexible and can learn any probability quite fast, but is therefore strongly dependent on the actual data. The honest prior performs good for learning probabilities that are honest (probabilities which lie in this case around 1/6). It depends less on the actual data which hinders the learning of probabilities that are substantially different from 1/6 (0.1666).

Conjugate prior

The mathematical details of the previous die example are easy to solve because we can compute the posterior distribution in closed form and use this to compute various quantities of interest, also in closed form. If we look more carefully, we see that our table data distribution and the Dirichlet prior play nicely together so that the posterior distribution is again a Dirichlet distribution. If we observe the data records one by one and update each time our current knowledge about the model parameters through Bayes' rule, we can keep storing our parameter knowledge in a single Dirichlet distribution. The form of the distribution stays the same, only the parameters of the distribution have to be updated. For the Dirichlet case, we have to update the count parameters. If the prior and data distribution are compatible in such a manner, we say that the prior is *conjugate* to the data model.

In addition to the table-Dirichlet conjugate pair, we will also use the Gaussian data model in combination with the Gaussian-Wishart prior. In this case, our data model is a multivariate Gaussian distribution. The parameters of this distribution are the mean vector and the covariance matrix. If we give the inverse covariance matrix (also called the precision matrix) a Wishart prior and the mean vector a multivariate Gaussian prior, we can again express the posterior distribution as a Gaussian-Wishart distribution with updated parameters. This will be used in Section 5.3.2 where we deal with linear-Gaussian Bayesian networks.

Informative and complexity-based priors

Until now we have just talked about prior distributions in general. For this thesis and more specifically for Chapter 4, it will be important to make a distinction between *uniform* and *improper* priors, *complexity*-based priors and *informative* priors.

Uniform and improper priors are those priors that take the same value or density for any model parametrization. Therefore, we can most of the time drop them from Bayes' rule since they are only a constant multiplication factor. The posterior distribution becomes the likelihood function, up to some multiplicative factor. This type of prior is a uniform distribution. If the parameters are continuous, we can only define such a prior if the parameters are known to lie within a bounded region. In the other case, we call it an improper prior because we cannot define a uniform distribution over such a space with a finite integral.

When working with parameter spaces with an infinite diameter, it is often useful not to use a improper prior, but some complexity-based prior. Such a prior does not contain any information about the problem at hand but is in general useful if this specific type of model class is used. A frequently used complexity-based prior tries to restrict the parameters from growing too large, but this is of course dependent on the actual data model. An example that will be discussed in detail in Chapter 6 are the multilayer perceptron models. Restricting their parameters makes sense from a theoretical point of view; multilayer perceptrons become more complex if their parameters grow. Such a complexity-based prior will prevent multilayer perceptrons from learning the data set by heart and therefore has a beneficial effect on the resulting models.

If on the other hand, our prior does contain useful information about the actual problem at hand, and is not useful for other applications of the same model class, we will call it an informative prior.

3.2 Classification

The main application of this thesis, introduced in Chapter 2, is a medical *classification* problem. Depending on the specific problem, there are a few different classes that observations can belong to. Although most practical problems are binary classification problems, dealing with more than two classes is not much different than handling the binary case. Because our problem is a binary classification problem, we will not clutter our notation and leave the multi-class case to the reader. We denote the two possible classes either with $\{0,1\}$, or with $\{\mathcal{C}_{\mathrm{N}},\mathcal{C}_{\mathrm{P}}\}$, where $\mathcal{C}_{\mathrm{N}}$ represents the negative classification and $\mathcal{C}_{\mathrm{P}}$ the positive classification.

We assume the existence of a *supervised* data set

$$\mathbf{D} = \{\boldsymbol{x}_k, t_k\}_{k=1}^{n},\ (\boldsymbol{x}_k, t_k) \in \mathbb{R}^v \times \{0,1\}.$$

Here, $\boldsymbol{x}_k$ is a vector of measurements for the k^{th} data entry, while t_k represents the corresponding class label. Once a model has been learned (its posterior distribution has been determined) using the prior knowledge and the data set, we can start using it.

3.2.1 Three levels of decision support

When we are presented with a classification problem, the primary goal is to predict the class label for each new observation. This can be done with the use of a binary decision function

$$\begin{aligned} \mathrm{g}(\,\cdot\,|\,\boldsymbol{\omega}\,) : \mathbb{R}^v &\longmapsto \{\mathcal{C}_{\mathrm{N}},\mathcal{C}_{\mathrm{P}}\} \\ \boldsymbol{x} &\longrightarrow \mathrm{g}(\,\boldsymbol{x}\,|\,\boldsymbol{\omega}\,). \end{aligned}$$

This decision function will *only* make a prediction for the class label. Often, more information is desired to make an actual decision. A higher level of decision support can be achieved with a model that predicts the $\mathcal{C}_{\mathrm{P}}$ class membership probability

$$\begin{aligned} \mathrm{f}(\,\cdot\,|\,\boldsymbol{\omega}\,) : \mathbb{R}^v &\longmapsto [0,1] \\ \boldsymbol{x} &\longrightarrow \mathrm{f}(\,\boldsymbol{x}\,|\,\boldsymbol{\omega}\,) = \mathrm{P}(\,\underline{t} = \mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x},\boldsymbol{\omega}\,). \end{aligned} \tag{3.5}$$

This higher level of decision support can also be reached by using a binary decision function in the Bayesian framework. Instead of using one fixed parametrization, we use now the posterior parameter distribution. Our binary decision function now becomes a random variable itself with a different Bernoulli distribution for each different input pattern $\boldsymbol{x}$. From this Bernoulli distribution, we can compute the required class membership probability.

We can reach an even higher level of decision support by using the class membership probability function described in Equation 3.5 in the Bayesian framework. Now, the class membership probability itself becomes a random variable. Instead of asking the model what its prediction is for the probability that a certain record belongs to class $\mathcal{C}_{\mathrm{P}}$, we get the complete uncertainty picture in the form of a probability density function for the Bernoulli parameter $\underline{\theta} = \mathrm{P}(\,\underline{t} = \mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x},\underline{\boldsymbol{\omega}}\,)$, for each record $\boldsymbol{x}$. Figure 3.5 shows these three levels of decision support for a binary classification problem.

3.2.2 Making a decision

We can step back from this highest level of decision support by marginalizing out the model parameters using that parameter distribution $\mathrm{p}(\boldsymbol{\omega})$ that denotes our

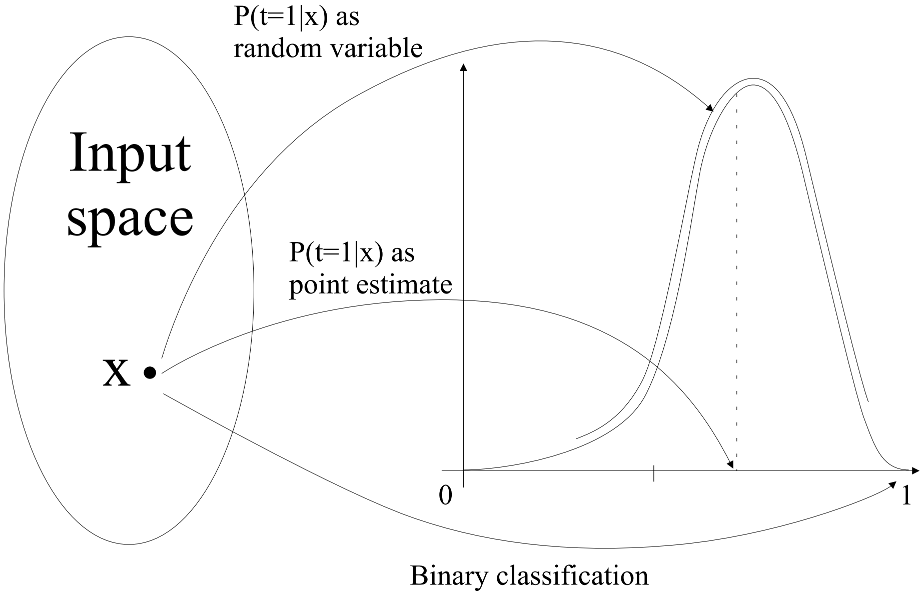


Figure 3.5: Three levels of decision support. The most basic decision support performs only a binary classification. A higher level of support indicates a point estimate of the probability that the subject $\underline{x}$ belongs to class 1. The highest level of decision support consists of the distribution of the different point estimates.

current knowledge of the problem. This will be most of the time the posterior distribution $\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D},\xi\,)$

$$\begin{aligned}
\mathrm{P}(\,\underline{t}=\mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x},\mathbf{D},\xi\,) &= \int_{\boldsymbol{\Omega}} \mathrm{P}(\,\underline{t}=\mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x},\boldsymbol{\omega}\,)\,\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D},\xi\,)\,d\boldsymbol{\omega} \\
&= \int_{[0,1]} \theta\,\mathrm{p}(\,\theta\,|\,\mathbf{D},\xi\,)\,d\theta \\
&= \mathrm{E}[\,\mathrm{P}(\,\underline{t}=\mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x},\underline{\boldsymbol{\omega}}\,)\,|\,\mathbf{D},\xi\,] \\
&= \mathrm{E}[\,\underline{\theta}\,|\,\mathbf{D},\xi\,].
\end{aligned}$$

In the second step, we changed the integration variable $\boldsymbol{\omega}$ to $\theta = \mathrm{P}(\,\underline{t} = \mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x},\boldsymbol{\omega}\,)$. Since this new parameter lies within the interval $[0,1]$, we have to change the integration region from $\boldsymbol{\Omega}$ to $[0,1]$, as is indicated in the formula. The symbol $\boldsymbol{\Omega}$ was the integration region of the model parameters $\boldsymbol{\omega}$.

Once we reach the class membership probability level, we have to use a threshold λ to reach an actual decision

$$\mathrm{g}_{\lambda}(\,\boldsymbol{x}\,|\,\mathbf{D},\xi\,) = \begin{cases} 1 & \text{if } \mathrm{E}[\,\mathrm{f}(\,\boldsymbol{x}\,|\,\boldsymbol{\omega}\,)\,|\,\mathbf{D},\xi\,] \geq \lambda, \\ 0 & \text{else.} \end{cases}$$

Choosing this threshold depends on the cost of making a wrong prediction and is therefore problem dependent. Based on a ROC curve, a sensible threshold value can often be found (see Section 3.3.2 for the introduction of the ROC curve).

A logistic regression example

We will demonstrate the three levels of decision support on an artificial classification problem to illustrate the difference between them and the difference with a maximum posterior model.

A data set was created that contains two variables and a binary class label (crosses and squares). The two covariates are generated from a Gaussian model where the mean depends on the class label. This data set is shown in Figure 3.6. We are interested in developing a model that is able to predict the probability that a certain observation (x_1, x_2) belongs to the square class.

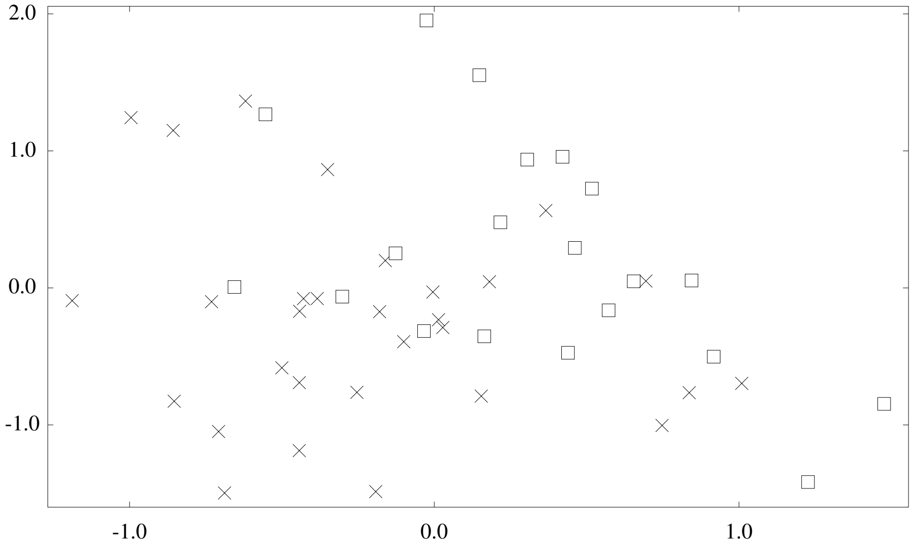


Figure 3.6: A two-dimensional artificial data set. The data set contains two classes which are indicated with crosses and squares respectively. The distribution within each class is Gaussian.

We choose a logistic regression model

$$\mathrm{f}(\, \underline{x} \,|\, \boldsymbol{\omega} \,) = \mathrm{logit}(\boldsymbol{\omega}_0 + \boldsymbol{\omega}_1 x_1 + \boldsymbol{\omega}_2 x_2) \tag{3.6}$$

as our class of membership probability models. The parameters of this model class are the coefficients $\boldsymbol{\omega}_0$, $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$. A logistic regression model is an extension of the well-known linear regression model; the linear combination of the covariates is transformed using the logit$(\cdot)$ function

$$\mathrm{logit}(y) = \frac{1}{1 + e^{-y}}.$$

The range of this function is the interval $]0,1[$, which allows us to interpret the output of a logistic regression model as a probability. In our case, we interpret it as the probability that observation $\underline{\boldsymbol{x}}$ belongs to the class corresponding to the squares. Because the logistic regression model is basically a transformed linear regression model, it decision boundary will still be linear. This means that the model can only separate classes using a linear hyper-plane. In our two-dimensional example, this will be a line. A more detailed explanation and motivation of the logistic regression model is presented in Section 6.2.3.

Now that we have met the data and chose the model class, we only have to specify our prior knowledge as a probability distribution over the model parameters $\boldsymbol{\omega}_0$, $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$. Since we have no idea how we have to choose these parameters before we observe the data set, we chose a non-informative complexity-based prior. This prior is a three-dimensional Gaussian distribution with independent components, zero mean and a standard deviation of 5.0:

$$\mathrm{p}(\,\boldsymbol{\omega}_i\,|\,\xi\,) \sim \mathcal{N}(0.0, 5.0).$$

Using Bayes' rule, we can compute the a posteriori distribution for our model parameters

$$\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D},\xi\,) \propto \mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)\,\mathrm{p}(\,\boldsymbol{\omega}\,|\,\xi\,).$$

Using this a posteriori distribution, we can compute the *maximum a posteriori* model. This is that logistic regression model that is specified with the maximum a posteriori model parameters $\boldsymbol{\omega}^*$,

$$\boldsymbol{\omega}^* = \mathrm{argmax}_{\boldsymbol{\omega}}(\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D},\xi\,)).$$

With this parametrized model, we can make a prediction that a new observation (y_1, y_2) belong to the square class:

$$\mathrm{P}(\,(y_1,y_2)\in\square\,|\,\boldsymbol{\omega}^*\,) = \mathrm{logit}(\boldsymbol{\omega}_0^* + \boldsymbol{\omega}_1^* y_1 + \boldsymbol{\omega}_2^* y_2).$$

An actual classification can be made with the use of a threshold λ, as explained in Section 3.2.1. Depending on the loss associated with misclassifying a cross or a square, we may adjust this threshold.

When working in the Bayesian framework, we are not looking for one "optimal" parametrization, as we did for the maximum a posteriori model, but use the a posteriori distribution instead. As such, we compute the probability that (y_1, y_2) belong to the square class as

$$\mathrm{P}(\,(y_1,y_2)\in\square\,|\,\mathbf{D},\xi\,) = \int \mathrm{P}(\,(y_1,y_2)\,|\,\boldsymbol{\omega}\,)\,\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D},\xi\,)\,d\boldsymbol{\omega}.$$

We have plotted both the maximum a posteriori and the Bayesian logistic regression model on Figure 3.7. The maximum a posteriori model is represented with the three dotted straight lines. The middle dotted line indicates the 50% decision boundary, while the other two lines indicate the 10% and 90% classification boundaries. Such a 10% or 90% classification boundary is the set of input vectors (y_1, y_2) that result is a classification $\mathrm{P}((y_1, y_2) \in \square) = 0.1$ or $\mathrm{P}((y_1, y_2) \in \square) = 0.9$. We plotted the same decision boundaries for the Bayesian model using dashed lines.

We see that these regions of equal class membership probabilities are straight lines for the maximum a posteriori logistic regression model, because of the linear part of Equation 3.6. The Bayesian model on the other hand has only a straight 50% boundary, which is as expected from two multivariate Gaussians that only differ by their means. The 10% and 90% boundaries are not straight anymore. The Bayesian model will be more conservative in its prediction when it has to make a prediction for input patterns further away from the gross of the data records. For records in the centre of the input pattern cloud, the Bayesian model can make bolder predictions. This is illustrated on Figure 3.7 with the black and white dot. In the case of the maximum a posteriori model, both these input patterns would get assigned a probability of 90% to belong to the cross-class. The Bayesian model will classify the black dot similarly with 90% probability to the cross-class, while its probability assignment for the white dot is lower than 90%. This behaviour is in line with our intuitive feeling.

The background of Figure 3.7 indicates the full Bayesian information level, where we used the variance of the predictions as uncertainty measure. The higher the variance, the darker the background will be. We see also that the uncertainty for the white dot prediction is higher than that for the black dot. Even on the 90% boundary of the Bayesian model, the uncertainty for input patterns further away from the pattern core will increase.

3.3 Performance measures

We will develop different models in this dissertation. To be able to compare all these models, some sort of *performance measure* is needed that reflects the usefulness of a model [38]. Although the Bayesian framework provides us with the model likelihood that can be used as performance measure [47], we do not find it suitable in the situation where completely different models with different types of priors have to be compared. When n different model classes $\mathcal{M}_1, \ldots, \mathcal{M}_n$ have to be considered, we can compute $\mathrm{P}(\,\mathcal{M}_i \,|\, \mathbf{D}, \xi\,)$ for each model class to know its probability. $\mathrm{P}(\,\mathcal{M}_i \,|\, \mathbf{D}, \xi\,)$ is called the model likelihood of the model class $\mathcal{M}_i$. To compute this term, we can directly apply Bayes' rule and condition on the model parameters $\boldsymbol{\omega}_i$ using the prior distribution

$$\begin{aligned} \mathrm{P}(\,\mathcal{M}_i \,|\, \mathbf{D}, \xi\,) &= \frac{\mathrm{p}(\,\mathbf{D} \,|\, \mathcal{M}_i, \xi\,)\, \mathrm{p}(\,\mathcal{M}_i \,|\, \xi\,)}{\mathrm{p}(\,\mathbf{D} \,|\, \xi\,)} \\ &\propto \int \mathrm{p}(\,\mathbf{D} \,|\, \mathcal{M}_i, \xi, \boldsymbol{\omega}_i\,)\, \mathrm{p}(\,\boldsymbol{\omega}_i \,|\, \mathcal{M}_i, \xi\,)\, d\boldsymbol{\omega}_i. \end{aligned} \tag{3.7}$$

We dropped the constant factor $\mathrm{p}(\,\mathbf{D} \,|\, \xi\,)$ from Equation 3.7. As we will see in Section 6.7, the integral from Equation 3.7 can be approximated using a Monte Carlo sum. Unfortunately, this integral will be hard to approximate because it is based on the prior distribution (see Section 6.7.1 for a detailed explanation). In essence, the support of the prior distribution is too large with respect to the support of $\mathrm{p}(\,\mathbf{D}\,|\,\mathcal{M}_i, \xi, \boldsymbol{\omega}_i\,)$, which results in a very slow convergence.

Besides this, we would like to compare the performance also against previous studies and want to have some natural feeling with the measure. Therefore we will look at a few performance measures specifically designed to evaluate classification models.

3.3.1 Misclassification rate

A first performance measure that pops up, is the mean *misclassification rate* because it has an intuitive interpretation. We assume that we have some kind of test $(T(\,\cdot\,|\,\boldsymbol{\omega}\,), \lambda)$, where $T(\,\cdot\,|\,\boldsymbol{\omega}\,)$ is a real-valued function for the possible input patterns, $\boldsymbol{\omega}$ are the parameters for this test function and $\lambda \in \mathbb{R}$ is a threshold. We denote the two possible classifications $\mathcal{C}_{\mathrm{P}}$ and $\mathcal{C}_{\mathrm{N}}$, which represents respectively the positive and the negative classification. We use our test to classify a patient with observations $\boldsymbol{x}$ as

$$\begin{array}{lcl} T(\,\boldsymbol{x}\,|\,\boldsymbol{\omega}\,) < \lambda & \Longrightarrow & \text{Classify as } \mathcal{C}_{\mathrm{N}} \\ T(\,\boldsymbol{x}\,|\,\boldsymbol{\omega}\,) \geq \lambda & \Longrightarrow & \text{Classify as } \mathcal{C}_{\mathrm{P}}. \end{array}$$

This classification depends on the threshold λ. How to choose this threshold is not always obvious. The models that will be developed in this thesis, will all be probabilistic models, which means that they attempt to model the probability that a patient with certain observations will belong to class $\mathcal{C}_{\mathrm{P}}$. If we use this *class probability* as our test function

$$T(\,\boldsymbol{x}\,|\,\boldsymbol{\omega}\,) = \mathrm{P}(\,\underline{t} = \mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x}, \boldsymbol{\omega}\,),$$

we have more insight in how to specify the threshold λ. If the loss of misclassifying a negative or a positive datum is the same, choosing $\lambda = 0.5$ will give us the optimal prediction. For making real predictions, we might want to take into account the different costs of making a *false positive* or *false negative* prediction. This means classifying a positive record as negative and vice versa.

But still, the misclassification rate has some unwanted behaviour; if the data set for which we compute this rate, contains only few samples, the misclassification rate can only take a few possible values. If, in addition to this, the prior distribution for the individual classes is unbalanced (as is the case for the ovarian tumour problem), the misclassification rate can take even fewer values. Another and more severe restriction that makes the usage of the misclassification rate less appropriate, is that it does not take into account the relative severity of different misclassifications. A misclassified object $\boldsymbol{x}$ where $T(\,\boldsymbol{x}\,|\,\boldsymbol{\omega}\,)$ lies just on the wrong side of the threshold λ has the same weight as an object that is classified in a completely wrong way. Finally, the misclassification rate depends on the actual threshold chosen, which makes is hard to compare different classification systems.

3.3.2 Area under the ROC curve

To deal with the problems of the misclassification rate, we will use the *area under the receiver operating characteristics curve (ROC)* [38], a performance measure that is often used to evaluate medical classification systems. To construct this curve, we need a few counts, all dependent on the threshold value λ:

True Positives ($\mathrm{TP}(\lambda)$) Number of *positive* observations that are *correctly* classified as $\mathcal{C}_\mathrm{P}$ by the test.

False Positives ($\mathrm{FP}(\lambda)$) Number of *negative* observations that are *wrongly* classified as $\mathcal{C}_\mathrm{P}$ by the test.

True Negatives ($\mathrm{TN}(\lambda)$) Number of *negative* observations that are *correctly* classified as $\mathcal{C}_\mathrm{N}$ by the test.

False Negatives ($\mathrm{FN}(\lambda)$) Number of *positive* observations that are *wrongly* classified as $\mathcal{C}_\mathrm{N}$ by the test.

Using these counts, we define the True Positive Rate, also called *sensitivity*, and the False Positive Rate, also called *1-specificity*, as

$$\begin{aligned}\mathrm{TPR}(\lambda) &= \frac{\mathrm{TP}(\lambda)}{\mathrm{TP}(\lambda)+\mathrm{FN}(\lambda)}\\ \mathrm{FPR}(\lambda) &= \frac{\mathrm{FP}(\lambda)}{\mathrm{FP}(\lambda)+\mathrm{TN}(\lambda)}.\end{aligned}$$

If we let the threshold vary from $-\infty$ to ∞ and plot the points ($\mathrm{FPR}(\lambda)$, $\mathrm{TPR}(\lambda)$), we get the receiver operating characteristics curve. This curve lies within the unit square $[0,1]^2$, starts from the upper right corner and ends in the lower left corner. We use the area under this curve as our performance measure. This area has a nice interpretation in terms of probabilities, related to the Wilcoxon statistic. The area is the probability that a negative record will be scored lower than a positive record according to the test function:

$$\mathrm{AUC} = \mathrm{P}(\, T(\,\boldsymbol{x}_\mathrm{N}\,|\,\boldsymbol{\omega}\,) < T(\,\boldsymbol{x}_\mathrm{P}\,|\,\boldsymbol{\omega}\,)\,|\,\boldsymbol{x}_\mathrm{N} \in \mathcal{C}_\mathrm{N} \text{ and } \boldsymbol{x}_\mathrm{P} \in \mathcal{C}_\mathrm{P}\,).$$

Figure 3.8 shows two typical ROC curves that correspond to a good test (the curve with the largest area) and a less good test (the curve with the smallest area). We see that if we choose our threshold in such a way that there is 85% chance to classify a positive patient as positive, that the test corresponding to the curve with the largest area (93.6) will perform better. This test will only classify 12.5% of the negative patients as positive, while the other test (area under the curve is 86.0) wrongly classifies 34.5% of the negative patients as positive.

3.4 To conclude

In this chapter, we introduced the principles of the Bayesian framework together with a short description of its history. We focussed on dealing with classification problems, which will be the main application of this dissertation. Different levels of decision support were indicated and different performance measures were introduced, allowing us to compare classification systems. We will use these performance measures to compare a whole range of classification models in Chapter 7.

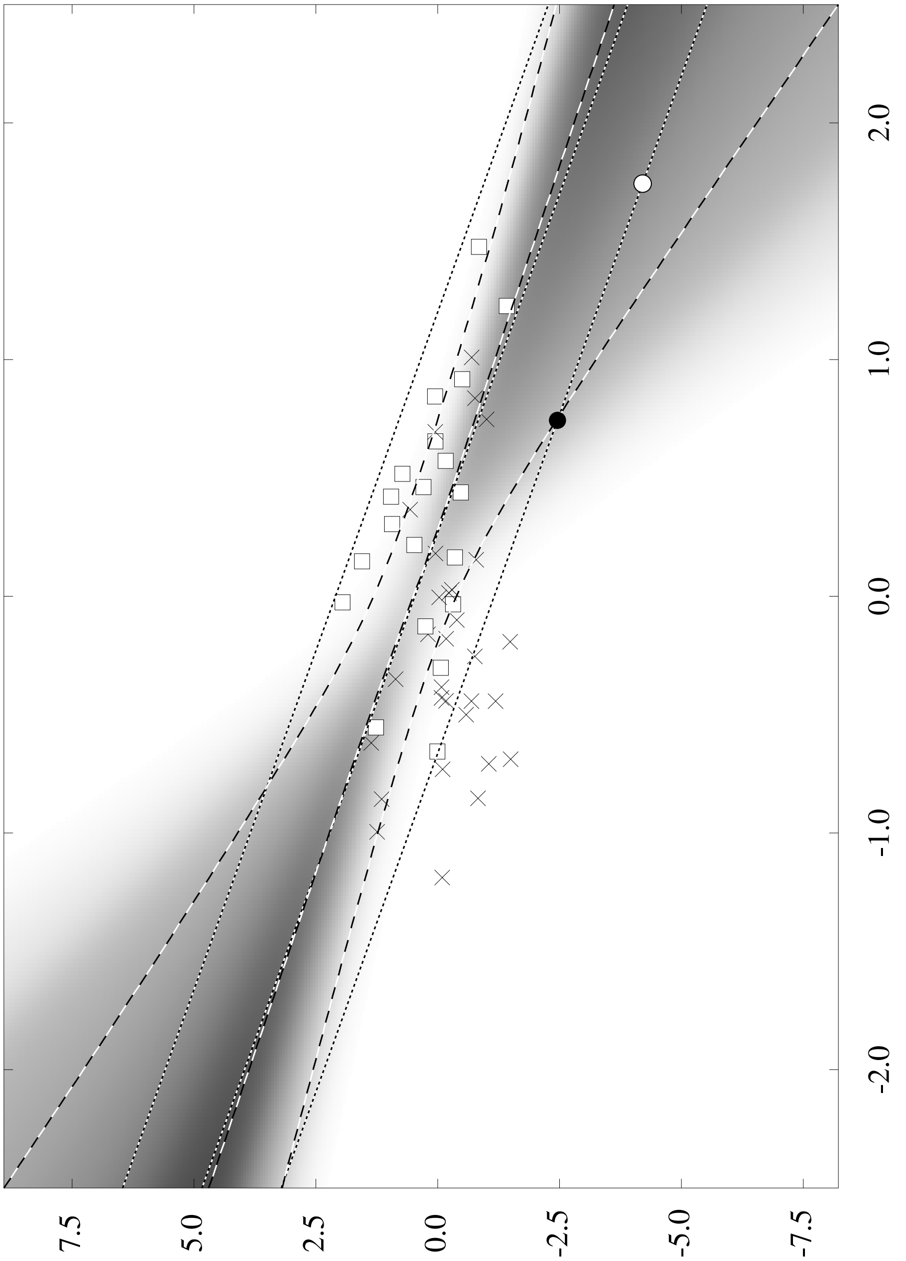


Figure 3.7: The Bayesian classification (dashed lines) and the maximum a posteriori classification (dotted lines). The lines indicate the 10%, 50% and 90% probability regions. The background indicates the full Bayesian information level measured with the variance. The darker the background, the higher the variance of the class membership probability.

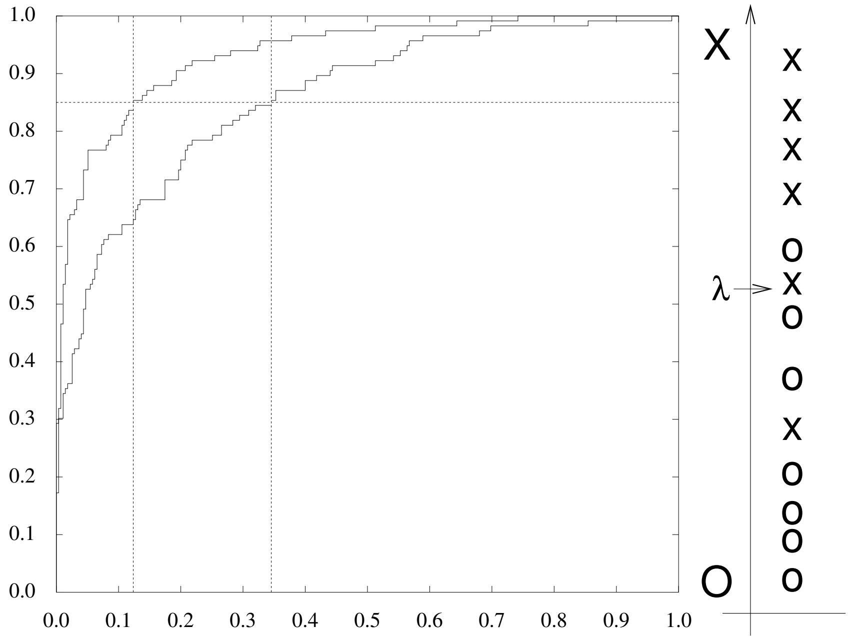
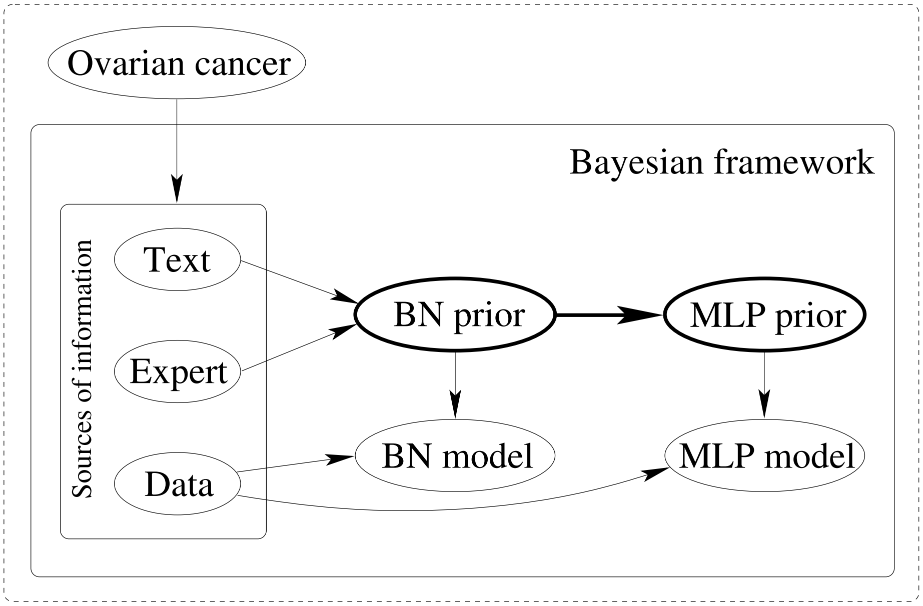


Figure 3.8: Two receiver operating characteristics curves representing the performance of two classification systems. The x axis represents the False Positive Rate, while the y axis indicates the True Positive Rate The larger the area under these curves, the better their corresponding tests will perform. The test corresponding to the large area (93.6) will classify 12.5% of the negative patients as positive while the other test corresponding to the smaller area (86.0) will classify 34.5% of the negative patients as positive when a sensitivity of 85% is required.

Chapter 4

Transformation of information representation



Often, we have some information about a problem, but this information cannot directly be included in the probabilistic model from where the actual observations are coming. This chapter will provide an outline how this information can still be used. At first, we choose a suitable model that can describe this information. Using a transformation *technique based on* virtual data sets*, we will be able to transform the information to a model class that is suitable to describe the actual observations also.*

4.1 An example

We will try to make things clear with a small example. Suppose we know that the diameter of the tomatoes produced by some plant have a uniform distribution between 8 and 10 centimetres. We represent this knowledge with the symbol ξ and will use this symbol from now on to indicate all the knowledge that is known besides the statistical data. This knowledge tells us something about the distribution of the diameter and can be written in symbols as

$$\mathrm{p}(\,\text{diameter}\,|\,\xi\,) = \begin{cases} \frac{1}{2} & \text{if diameter} \in [8, 10] \text{ cm} \\ 0 & \text{else.} \end{cases}$$

Since the tomato market is more oriented towards the *weight* of a tomato instead of its *diameter*, we would like to transform our knowledge ξ about the diameter to a distribution describing the same knowledge in a model for the weight of our tomatoes. In such an easy case as this, we can compute the resulting distribution directly using the *change of variables* formula

$$\mathrm{p}(\,\omega\,|\,\xi\,) = \mathrm{p}(\,\theta(\omega)\,|\,\xi\,) \times |\det(\frac{\partial\omega}{\partial\theta})|^{-1}. \tag{4.1}$$

This formula allows us to change the variable θ (the diameter) to the new variable ω (the weight) if the transformation $\omega(\theta)$ is known, differentiable and bijective. The function $\theta(\omega)$ represents the inverse function of $\omega(\theta)$ and $|\det(\frac{\partial\omega}{\partial\theta})|$ is called the Jacobian of the transformation.

We assume that the weight of a tomato can be computed as its volume times a constant factor $\mathrm{C}^{\underline{\mathrm{st}}}$ representing the weight per volume of tomatoes

$$\begin{aligned} \text{weight} &= \mathrm{C}^{\underline{\mathrm{st}}} \times \text{volume} \\ &= \mathrm{C}^{\underline{\mathrm{st}}} \times \frac{\pi}{6}\text{diameter}^3 \\ \text{diameter} &= \left(\frac{6\text{ weight}}{\mathrm{C}^{\underline{\mathrm{st}}}\pi}\right)^{1/3} \\ \frac{\partial\text{weight}}{\partial\text{diameter}} &= \frac{\mathrm{C}^{\underline{\mathrm{st}}}\pi}{2}\text{diameter}^2 \\ &= \frac{2}{(\mathrm{C}^{\underline{\mathrm{st}}}\pi)^{1/3}(6\text{ weight})^{2/3}}. \end{aligned}$$

This results into the transformed distribution

$$\mathrm{p}(\,\text{weight}\,|\,\xi\,) = \frac{1}{\mathrm{C}^{\underline{\mathrm{st}}}\pi^{1/3}(6\text{ weight})^{2/3}}, \quad \text{if weight} \in \left[256\frac{\mathrm{C}^{\underline{\mathrm{st}}}\pi}{3}, 500\frac{\mathrm{C}^{\underline{\mathrm{st}}}\pi}{3}\right].$$

If we now weigh a series of tomatoes, we can then easily update our *prior* knowledge $\mathrm{p}(\,\text{weight}\,|\,\xi\,)$ with measurements of these tomatoes. The resulting posterior distribution represents now our belief about the distribution of the weight.

Such an ideal case as above will not occur often in practice; the transformation of our knowledge ξ can most of the time not be computed exactly. We will have to find others means for transforming our information representation.

4.2 Donor models

The first type of model, which will be called a *donor* model, will describe a joint distribution and has the capability to *incorporate prior information.* For instance, experience and knowledge of a domain expert could be expressed into this model via the usage of a prior distribution $\mathrm{p}(\,\theta\,|\,\xi\,)$. Here, θ represents the parameters of the donor model class that the expert found suitable to express his knowledge ξ. Other valuable information that we will encounter is information in some specific format (e.g., textual information) that could be used to specify this model. We represent this knowledge with the symbol ξ.

In addition to the advantageous property of handling background information well, we assume that this model is not very suitable to update this *informative* prior distribution $\mathrm{p}(\,\theta\,|\,\xi\,)$ to its posterior $\mathrm{p}(\,\theta\,|\,\mathbf{D}, \xi\,)$ by using the data $\mathbf{D}$. If this *was* the case, it would be optimal to combine the data directly into this donor model, instead of using a different model class and transforming the information representation to there.

In our case, our donor model will be a Bayesian network, as explained in Chapter 5. The parameters of this model have a clear probabilistic interpretation, providing a domain expert with a tool to express his knowledge. In addition to this, we will be able to derive a prior for its structure based on vast quantities of electronic literature. On the other side, a Bayesian network that is defined by an expert has several drawbacks. Its structure is often too restricted to model the real data: usually it is discrete in nature, suffers from a huge number of parameters that have to be determined from the data and models a joint distribution of the whole domain instead of the conditional distribution we are interested in to perform classification. Although it does not harm to model the whole domain using a joint distribution, we would like the model to concentrate specifically on modelling the distribution of the class labels.

4.3 Acceptor models

The opposite of a donor model, which we will call an *acceptor* model, is a model that has generally poor capabilities to incorporate prior knowledge. In fact, most of the time, no sensible prior or only a complexity-based prior $\mathrm{p}(\omega)$ can be defined directly. We assume that it is not possible to compute a direct transformation by using the change of variables formula Equation 4.1. If you can compute this direct transformation, as was the case for our tomato example, you are lucky and there is no need to go through the procedures that are described next. Note also that it will not be necessary that this model describes a joint distribution.

As acceptor model class, we select that model class that has to tackle our problem, and we require that it has good learning capabilities from data. It should have the necessary power and flexibility to describe the data and its parametrization must be continuous. This last property means that a small change in the model parameters should result in a small change of the resulting joint or conditional distribution. This allows us to even use a slightly wrong prior.

We will use a multilayer perceptron model as an acceptor model, and will introduce this model in Chapter 6. Its parameters have no clear interpretation, leaving it virtually impossible to specify prior knowledge directly into a prior distribution other than complexity-related information. It will model a continuous conditional distribution, specifically suitable for classification, our problem at hand. In addition to this, multilayer perceptrons are known to have good learning capabilities from data because of their relatively small number of parameters that are needed to express a wide range of different distributions.

We use the donor/acceptor terminology that was developed by Radford Neal [64]. Note however that the same procedure as Neal describes was developed and published by the author [5] beforehand.

4.4 Transformation of representation

Both models have their own strong points, which are more or less complementary, as indicated in Figure 4.1. The donor model is suitable to describe the prior information, but fails to model the problem at hand very well. The acceptor model on the other hand is not suitable to use as a tool for gathering prior information, but has good learning performance or is the specific tool we need for solving our task.

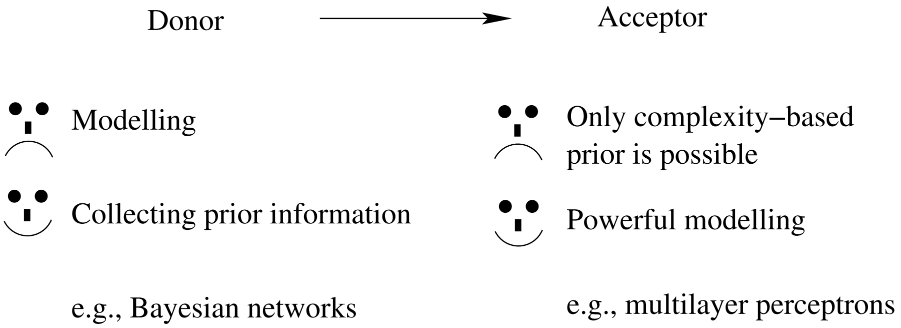

Donor
Acceptor
Modelling
Only complexity–based prior is possible
Collecting prior information
Powerful modelling
e.g., Bayesian networks
e.g., multilayer perceptrons

Figure 4.1: The donor model (left) is used to collect the prior knowledge. Because the donor model class is not suitable to learn from data, the collected prior knowledge is transformed to an acceptor model. This acceptor model will be able to update the transformed prior knowledge to a posterior distribution with the use of the data. The acceptor model is not suitable to directly specify the prior knowledge, so we have to go through the whole process.

It is therefore natural to seek the combination of both. Stated in the Bayesian framework, we are looking for a prior distribution for the parameters ω of the acceptor model that contains the information that was captured using the donor model.

We can write out this informative prior distribution $\mathrm{p}(\,\omega\,|\,\xi\,)$ by conditioning on all possible data sets $\mathbf{D}_k$ containing k records and the donor model parameters θ:

$$\begin{aligned}\mathrm{p}(\,\omega\,|\,\xi\,) &= \sum_{\mathbf{D}_k} \mathrm{p}(\,\omega\,|\,\mathbf{D}_k, \xi\,)\,\mathrm{p}(\,\mathbf{D}_k\,|\,\xi\,) \\ &= \sum_{\mathbf{D}_k} \mathrm{p}(\,\omega\,|\,\mathbf{D}_k, \xi\,) \int_\theta \mathrm{p}(\,\mathbf{D}_k\,|\,\theta, \xi\,)\,\mathrm{p}(\,\theta\,|\,\xi\,)\,d\theta \\ &\approx \sum_{\mathbf{D}_k} \mathrm{p}(\,\omega\,|\,\mathbf{D}_k, \xi_\mathrm{c}\,) \int_\theta \mathrm{p}(\,\mathbf{D}_k\,|\,\theta, \xi\,)\,\mathrm{p}(\,\theta\,|\,\xi\,)\,d\theta \qquad (4.2)\end{aligned}$$

The only approximation we make in Equation 4.2 is that we assume the data set $\mathbf{D}_k$ to be large enough to contain sufficient information so we can drop the informative knowledge ξ from the term $\mathrm{p}(\,\omega\,|\,\mathbf{D}_k, \xi\,)$. We exchange this knowledge term ξ with only the complexity-based knowledge ξ_c. Therefore, this formula still contains the non-informative prior $\mathrm{p}(\omega)$ that we were able to define for the acceptor model and updates this to an informative prior.

Although this formula does not give us an easy analytic form to work with, it can be read directly into a procedure to draw parametrizations from $\mathrm{p}(\,\omega\,|\,\xi\,)$:

1. Generate a donor parametrization θ from the informative prior $\mathrm{p}(\,\theta\,|\,\xi\,)$.

2. Generate a data set $\mathbf{D}_k$ from the joint distribution defined by the donor model with parameters θ.

3. Generate a parametrization ω from the a posteriori distribution $\mathrm{p}(\,\omega\,|\,\mathbf{D}_k, \xi_\mathrm{c}\,)$.

If an analytic form is necessary, we can draw several parametrizations from $\mathrm{p}(\,\omega\,|\,\xi\,)$ and estimate this distribution using some general class of probability distributions.

4.4.1 Choosing the data set size

An important parameter in this transformation is the size k of the virtual data sets $\mathbf{D}_k$. The size should be large enough to allow us to exchange ξ from the term $\mathrm{p}(\,\omega\,|\,\mathbf{D}_k, \xi\,)$ with the complexity-based knowledge ξ_c. The larger we choose k, the more accurate the transformation will be, although it can be dangerous to set it too large; the larger k gets, the more peaked $\mathrm{p}(\,\omega\,|\,\mathbf{D}_k\,)$ will get. If none of the data sets $\mathbf{D}_k$ are like the real data set, none of the posterior distributions $\mathrm{p}(\,\omega\,|\,\mathbf{D}_k\,)$ will have the optimal acceptor model parametrization in their support, resulting in a prior $\mathrm{p}(\,\omega\,|\,\xi\,)$ that does not even contain the optimal parametrization. By choosing k not too large, $\mathrm{p}(\,\omega\,|\,\xi\,)$ will be the weighted sum of not-too-peaked posterior distributions, resulting in a usable prior even if $\mathrm{p}(\,\theta\,|\,\xi\,)$ is somehow wrong.

4.4.2 Discrete and continuous variables

It is often easier to think and reason about discrete distributions instead of probability density functions. Especially in medicine there is a habit to discretize variables in a few bins and define distributions only using these bins. By doing so, some information is lost, but insight in the problem can be gained.

In our test case, things are not different. It was much more natural for our expert to specify the relation between the variables in a discrete setting instead of using some class of continuous distributions. Because our neural network model works most naturally with continuous variables and some domain variables are continuous, it would be artificial to enforce the network to use only the discretized version of the data.

This requirement results in an extended formulation for $\mathrm{p}(\,\omega\,|\,\xi\,)$, where $\mathrm{p}(\,\mathbf{D}_\mathrm{c}\,|\,\mathbf{D}_\mathrm{d}\,)$ indicates how we have to generate a continuous data set $\mathbf{D}_\mathrm{c}$ from a discrete one $\mathbf{D}_\mathrm{d}$

$$\begin{aligned}
\mathrm{p}(\,\omega\,|\,\xi\,) &= \int_{\mathbf{D}_\mathrm{c}} \mathrm{p}(\,\omega\,|\,\mathbf{D}_\mathrm{c}, \xi\,)\,\mathrm{p}(\,\mathbf{D}_\mathrm{c}\,|\,\xi\,)\,d\mathbf{D}_\mathrm{c} \\
&\approx \int_{\mathbf{D}_\mathrm{c}} \mathrm{p}(\,\omega\,|\,\mathbf{D}_\mathrm{c}, \xi_\mathrm{c}\,)\,\mathrm{p}(\,\mathbf{D}_\mathrm{c}\,|\,\xi\,)\,d\mathbf{D}_\mathrm{c} \\
&= \int_{\mathbf{D}_\mathrm{c}} \mathrm{p}(\,\omega\,|\,\mathbf{D}_\mathrm{c}, \xi_\mathrm{c}\,) \sum_{\mathbf{D}_\mathrm{d}} \mathrm{p}(\,\mathbf{D}_\mathrm{c}\,|\,\mathbf{D}_\mathrm{d}, \xi\,)\,\mathrm{p}(\,\mathbf{D}_\mathrm{d}\,|\,\xi\,)\,d\mathbf{D}_\mathrm{c} \\
&= \int_{\mathbf{D}_\mathrm{c}} \mathrm{p}(\,\omega\,|\,\mathbf{D}_\mathrm{c}, \xi_\mathrm{c}\,) \sum_{\mathbf{D}_\mathrm{d}} \mathrm{p}(\,\mathbf{D}_\mathrm{c}\,|\,\mathbf{D}_\mathrm{d}, \xi\,) \int_\theta \mathrm{p}(\,\mathbf{D}_\mathrm{d}\,|\,\theta, \xi\,)\,\mathrm{p}(\,\theta\,|\,\xi\,)\,d\theta\,d\mathbf{D}_\mathrm{c} \\
&= \int_{\mathbf{D}_\mathrm{c}} \mathrm{p}(\,\omega\,|\,\mathbf{D}_\mathrm{c}, \xi_\mathrm{c}\,) \sum_{\mathbf{D}_\mathrm{d}} \mathrm{p}(\,\mathbf{D}_\mathrm{c}\,|\,\mathbf{D}_\mathrm{d}\,) \int_\theta \mathrm{p}(\,\mathbf{D}_\mathrm{d}\,|\,\theta\,)\,\mathrm{p}(\,\theta\,|\,\xi\,)\,d\theta\,d\mathbf{D}_\mathrm{c}.
\end{aligned}$$

The only thing that changed with respect to the previous formula, is the conditioning on the discrete data sets and the generation of these discrete data sets with the donor model. The approximation is due to the exchange of ξ with ξ_c, as explained above.

The procedure to generate ω from $\mathrm{p}(\,\omega\,|\,\xi\,)$ is the same as previously defined, except that we have to generate a continuous data set from the discrete one. We treat the continuous records and variables as being independent given the discrete values

$$\mathrm{p}(\,\mathbf{D}_\mathrm{c}\,|\,\mathbf{D}_\mathrm{d}\,) = \prod_{i=1}^{n} \prod_{j=1}^{v} \mathrm{p}(\,\underline{x}_\mathrm{c}^{ij}\,|\,\underline{x}_\mathrm{d}^{ij}\,),$$

where $\underline{x}^{ij}$ is the value of the j^th variable in the i^th data record.

Several strategies can be adopted for defining $\mathrm{p}(\,\underline{x}_\mathrm{c}\,|\,\underline{x}_\mathrm{d}\,)$ using the original discretization bins. The discretization of a variable is a set of disjunct intervals $\{[a_i, b_i[\}_{i=1}^{d}$ that have been used to discretize the data in the first place. The random variable $\underline{x}$ should lie within the union $\bigcup_{i=1}^{d}[a_i, b_i[$.

Figure 4.2 displays a discrete distribution over five bins where the probability of each bin is indicated by the *length* of the lines with the circles under the x

axis. The indications above the x axis represent the different possibilities to construct a continuous distribution from our discrete one.

Middle of a bin

For this method, we define $\mathrm{p}(\,\underline{x}_\mathrm{c}\,|\,\underline{x}_\mathrm{d} = i\,) \equiv \delta_{(a_i+b_i)/2}(\underline{x}_\mathrm{c})$, which sets $\underline{x}_\mathrm{c}$ fixed to the middle of the i^th bin $[a_i, b_i[$. We denote with $\delta_y(x)$ the Dirac distribution around y:

$$\int f(x)\,\delta_y(x)\,dx = f(y).$$

The discrete nature of this method will result in bad generalization power as $\underline{x}_\mathrm{c}$ will take only d values. This distribution is indicated with the vertical lines with the squares on Figure 4.2. The *height* of such a line represent the volume of the Dirac peak.

Uniform over a bin

To deal with the above problem, we could choose $\underline{x}_\mathrm{c}$ uniform from its corresponding bin. This will solve half the problem: the probability density function of $\underline{x}_\mathrm{c}$ is now a stepwise function with discontinuities at the borders of the bins. These discontinuities result in large network parametrizations for the same reason as explained in Section 6.3.1. Note that this density function is *not* the same as the discrete distribution, because the width of a bin is now taken into account.

Some overlap between bins

We can get rid of the discontinuities by using a continuous distribution for each bin. A Gaussian distribution centred around the middle of the bin with a certain percentage overlap with the next bin is a suitable choice. These separate weighted Gaussian distributions are represented by the dotted curves in Figure 4.2, while their sum, the resulting density function, is indicated by the full line. The percentage of overlap here was 15%, meaning that 0.15 of the probability mass of one such Gaussian falls beyond the boundaries of the corresponding bin. This is achieved by choosing the standard deviation of each Gaussian as 0.347 times the width of the corresponding bin.

4.5 Other approaches

Many other approaches to combine prior domain knowledge and data build on learning theories [25, 12, 86] that made it possible to formalize how the incorporation of domain knowledge reduces the statistical complexity of learning for inductive techniques, both in the classical statistical context [36, 39] or in the Bayesian context [40].

On the practical side, Abu-Mostafa [1] and Niyogi et al. [66] reported methods for exploiting certain regularities and symmetries of the input space that

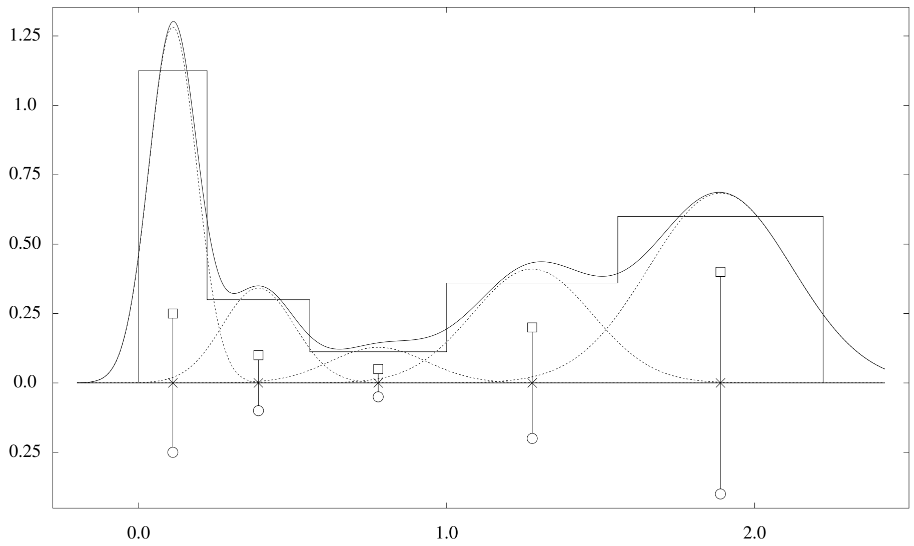


Figure 4.2: A discrete distribution and some possible methods to convert it to a continuous distribution. The discrete distribution is indicated under the x axis by the lengths of the lines and the circles. Above the x axis is a sum of Dirac distributions, indicated by the vertical lines and the squares, a stepwise density and a sum of Gaussians. The dotted lines denote the Gaussian distributions for each discretization bin.

are known a priori to enlarge the data set. Another approach — the knowledge-based artificial neural network — uses the prior knowledge for selecting an appropriate multilayer perceptron architecture [85]. Other methods focussed on incorporating prior knowledge into neural networks include the work of Sowmya Ramachandran [71] who basically suggests to build Bayesian networks with knowledge- and neural network based local models.

We will discuss one intuitive approach in more detail; suppose that our prior information ξ is a data set $\mathbf{D}_\mathrm{p}$. In this case, its seems easy to specify the posterior parameter distribution for the acceptor model

$$\mathrm{p}(\,\omega\,|\,\mathbf{D},\xi\,) = \mathrm{p}(\,\omega\,|\,\mathbf{D},\mathbf{D}_\mathrm{p}\,);$$

we simply have to concatenate both data sets, and use this large data set to compute the posterior distribution.

In practice, this is not always the optimal approach. Often, the prior data set $\mathbf{D}_\mathrm{p}$ comes from a slightly different distribution, it can be older, or have missing fields. Combining the data sets is only valid when they have the same distribution. Still, this prior data set contains valuable information that we would like to use.

A possible solution to use this prior data set $\mathbf{D}_\mathrm{p}$ is to define a prior distribution over the acceptor parameter space, and update this prior to a posterior using the real data set $\mathbf{D}$. We still make the mistake of combining information from different distributions, but this time the additional step of generating an a priori distribution gives us extra possibilities: it is unlikely that the parametrization ω^* that generated $\mathbf{D}$ will be the maximum a priori parametrization $\text{argmax}_\omega(\mathrm{p}(\,\omega\,|\,\mathbf{D}_\mathrm{p}\,))$, but it is reasonable to expect that ω^* will still have a reasonable probability if the parametrizations of the acceptor model are continuous. With this last property, we mean that two models with slightly different parameters will result in probability distributions that are only slightly different. If we suspect that the prior data does not follow the same distribution as the real data set, we can estimate the informative a priori distribution using some parametric distribution $\mathrm{p}(\,\omega\,|\,\boldsymbol{\nu}\,)$:

$$\mathrm{p}(\,\omega\,|\,\boldsymbol{\nu}\,) \approx \mathrm{p}(\,\omega\,|\,\mathbf{D}_\mathrm{p}\,),$$

and make it wider by increasing its variance. This way, we will use the information from the prior data set, but do not believe it that strictly. This is something we cannot do by directly combining the two data sets.

We will illustrate the above with an extreme example. Suppose that we want to perform a regression task where we know that the expected mean of $\underline{y}$ given $\underline{x}$ is based on the sine function:

$$\mathrm{E}[\,\underline{y}\,|\,\underline{x}\,] = \omega_1\,\sin(\omega_2\,\underline{x} + \omega_3).$$

Both the real data set $\mathbf{D}$ and the prior data set $\mathbf{D}_\mathrm{p}$ are generated from such a model, but with slightly different parameter setting. Both data sets are shown on Figure 4.3, the real data set is indicated using crosses, while the prior data set is indicated using circles. A first model is trained using the posterior distribution based on concatenating both data sets together. Its conditional mean is indicated using a dashed line. Another model was trained by first estimating the prior distribution given $\mathbf{D}_\mathrm{p}$. This a priori distribution was then converted to an a posteriori distribution using the real data $\mathbf{D}$. The conditional mean of this model is represented using a full line.

Figure 4.3 clearly indicates that estimating the prior distribution has a beneficial effect over simply combining both data sets in this case. Although the difference depends on the problem and the acceptor model, we will choose the approach where the prior information is used to estimate a prior distribution in the acceptor model parameter space.

Note that we can extend the above discussion to a more general case; instead of assuming that our prior information consists of one data set $\mathbf{D}_\mathrm{p}$, we can assume that our prior information ξ is given as a distribution over the model parameters θ of a certain donor model, as was the case in Section 4.4. Computing the posterior directly corresponds to the above procedure where the two data sets are combined directly:

$$\begin{aligned}
\mathrm{p}(\,\omega\,|\,\mathbf{D},\xi\,) &= \sum_{\mathbf{D}_k} \mathrm{p}(\,\omega\,|\,\mathbf{D},\mathbf{D}_k,\xi\,)\,\mathrm{p}(\,\mathbf{D}_k\,|\,\mathbf{D},\xi\,) \\
&= \sum_{\mathbf{D}_k} \mathrm{p}(\,\omega\,|\,\mathbf{D},\mathbf{D}_k,\xi\,) \int_\theta \mathrm{p}(\,\mathbf{D}_k\,|\,\theta,\xi\,)\,\mathrm{p}(\,\theta\,|\,\mathbf{D},\xi\,)\,d\theta \\
&\approx \sum_{\mathbf{D}_k} \mathrm{p}(\,\omega\,|\,\mathbf{D},\mathbf{D}_k,\xi_{\mathrm{c}}\,) \int_\theta \mathrm{p}(\,\mathbf{D}_k\,|\,\theta\,)\,\mathrm{p}(\,\theta\,|\,\mathbf{D},\xi\,)\,d\theta.
\end{aligned}$$

The same reasoning as above suggests that a direct combination of $\mathbf{D}$ and $\mathbf{D}_k$ can result in suboptimal performance.

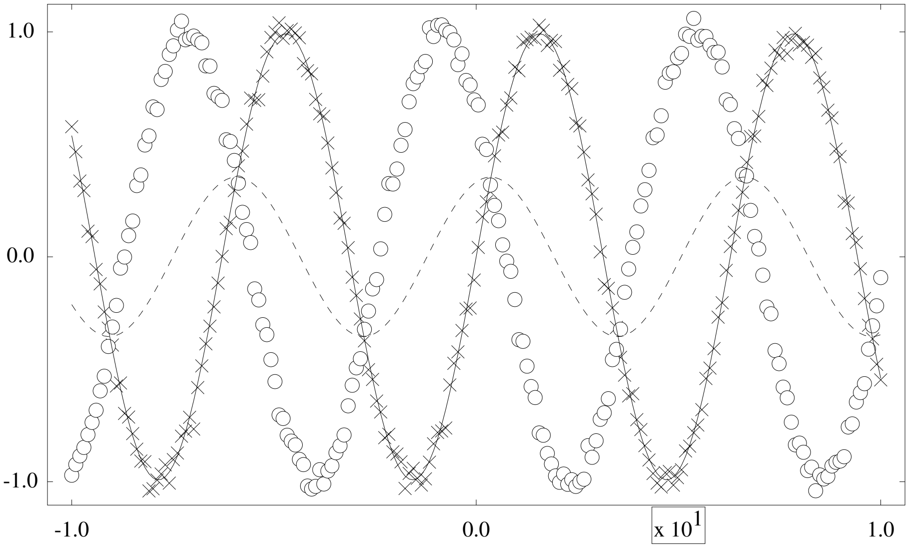


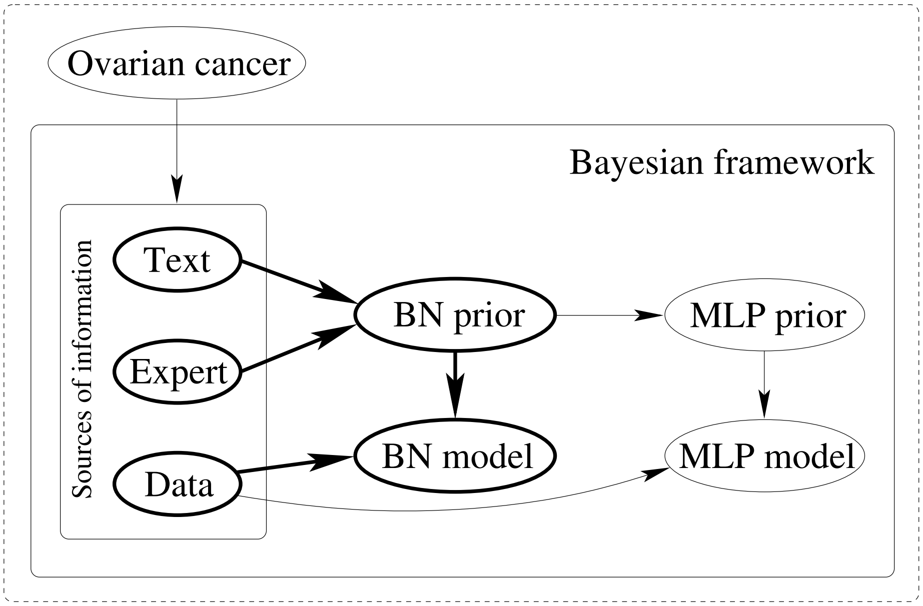
Figure 4.3: The real data set is indicated using crosses while the prior data set is indicated using circles. A first model is shown where the posterior distribution is computed by concatenating both data sets (dashed line). The full line indicates the conditional mean of another model where the posterior distribution was computed by first estimating the prior distribution given the prior data set $\mathbf{D}_{\mathrm{p}}$. This prior distribution was then updated to a posterior distribution using the real data set.

4.6 To conclude

We introduced a technique to transform the information representation from one model to another by using virtual data sets. This technique was developed by the author in close cooperation with Peter Antal [5, 3, 4]. Although this method is introduced here in a general way, it was developed with a specific purpose in mind; the application specific goal is to combine the different types of information available for the ovarian tumour problem introduced in Section 2. This will be accomplished using the above transformation technique in combination with Bayesian networks and multilayer perceptrons. These two models will be introduced in the following two chapters.

Chapter 5

Bayesian networks



In the previous chapter, we discussed a technique for transforming the information representation from one model to another. A donor model was used to gather the prior knowledge in the form of an informative prior distribution. Once this prior was specified, this information was transformed to a so-called acceptor model via the use of virtual data sets.

This chapter will introduce and discuss the donor model class that we will use for the ovarian classification problem introduced in Chapter 2. We selected the donor model class as the class of Bayesian networks*, also called* belief networks*, for various reasons. These models belong to the class of graphical models because* *they can be represented by a graph where each node represents a domain variable.*

We will describe the model, its usage, and some of the more important algorithms that exist to learn and use this model.

5.1 Description

A Bayesian network is a probabilistic model that describes a joint distribution over a set of random variables, also called the domain variables [78, 15, 52, 9, 35]. If we suppose we have v domain variables $(\underline{x}_1, \ldots, \underline{x}_v)$, we can always apply the chain rule of probability:

$$\begin{aligned} \mathrm{p}(\underline{x}_1, \ldots, \underline{x}_v) &= \mathrm{p}(\underline{x}_1)\,\mathrm{p}(\,\underline{x}_2 \,|\, \underline{x}_1\,) \cdots \mathrm{p}(\,\underline{x}_v \,|\, \underline{x}_1, \ldots, \underline{x}_{v-1}\,) \\ &= \prod_{i=1}^{v} \mathrm{p}(\,\underline{x}_i \,|\, \underline{x}_1, \ldots, \underline{x}_{i-1}\,). \end{aligned} \tag{5.1}$$

This decomposition of the joint distribution into v factors is dependent on the *order* of the variables for which we apply the chain rule of probability and hereby will have an impact on the construction of Bayesian networks.

Starting from Equation 5.1, we try to simplify the factors one by one, depending on the problem — the joint distribution — we want to model; suppose that $\underline{x}_3$ is conditionally independent of $\underline{x}_1$ given $\underline{x}_2$. In other words, the additional knowledge of $\underline{x}_1$ does not learn us anything new about $\underline{x}_3$ if we already knew $\underline{x}_2$. We denote this property with $(\,\underline{x}_3 \perp \underline{x}_1 \,|\, \underline{x}_2\,)$. This means that we can simplify the third factor to

$$\mathrm{p}(\,\underline{x}_3 \,|\, \underline{x}_1, \underline{x}_2\,) = \mathrm{p}(\,\underline{x}_3 \,|\, \underline{x}_2\,).$$

If we make these simplifications for each node $\underline{x}_i$, we get

$$\begin{aligned} \mathrm{p}(\underline{x}_1, \ldots, \underline{x}_v) &= \prod_{i=1}^{v} \mathrm{p}(\,\underline{x}_i \,|\, \pi(\underline{x}_i)\,) \\ \pi(\underline{x}_i) &\subset \{\underline{x}_1, \ldots, \underline{x}_{i-1}\}, \end{aligned} \tag{5.2}$$

where we call the set $\pi(\underline{x}_i)$ the *parents* of node $\underline{x}_i$.

This formulation of the joint probability distribution contains the *bilateral* nature of a Bayesian network: at first, we simplify the joint probability distribution by acting on each factor of Equation 5.1 separately, ending up with a sparse representation. Next, we have to specify the univariate conditional distributions $\mathrm{p}(\,\underline{x}_i \,|\, \pi(\underline{x}_i)\,)$ that are left. The former is called the *structure* of the Bayesian network, while the distributions in the latter are called the *local dependency models.*

5.1.1 Structure and conditional independency

By carrying out the simplifications in Equation 5.2, we construct an easier representation of our joint probability distribution. This can be represented by using a *directed acyclic graph* where the nodes correspond to the domain variables and all the incoming edges of a node come from the parents. This directed acyclic graph represents the *structure* of the Bayesian network. Figure 5.1 displays two such structures for a five variable domain. The fully connected graph (indicated with $\mathcal{S}_{\mathrm{bn}}^{\mathrm{C}}$) corresponds to a domain where no simplifications can be made, while the other graph corresponds to the simplifications $(\,\underline{b} \perp \underline{a}\,|\,\phi\,)$, $(\,\underline{c} \perp \underline{a},\underline{b}\,|\,\phi\,)$, $(\,\underline{d} \perp \underline{c}\,|\,\underline{a},\underline{b}\,)$ and $(\,\underline{e} \perp \underline{a},\underline{b}\,|\,\underline{c},\underline{d}\,)$. The symbol ϕ denotes the empty set; $(\,\underline{b} \perp \underline{a}\,|\,\phi\,)$ therefore indicates that $\underline{a}$ is independent of $\underline{b}$. The joint distribution in this case can thus be simplified as

$$\begin{aligned}\mathrm{p}(\underline{a},\underline{b},\underline{c},\underline{d},\underline{e}) &= \mathrm{p}(\underline{a})\,\mathrm{p}(\,\underline{b}\,|\,\underline{a}\,)\,\mathrm{p}(\,\underline{c}\,|\,\underline{a},\underline{b}\,)\,\mathrm{p}(\,\underline{d}\,|\,\underline{a},\underline{b},\underline{c}\,)\,\mathrm{p}(\,\underline{e}\,|\,\underline{a},\underline{b},\underline{c},\underline{d}\,)\\ &= \mathrm{p}(\underline{a})\,\mathrm{p}(\underline{b})\,\mathrm{p}(\underline{c})\,\mathrm{p}(\,\underline{d}\,|\,\underline{a},\underline{b}\,)\,\mathrm{p}(\,\underline{e}\,|\,\underline{c},\underline{d}\,).\end{aligned}$$

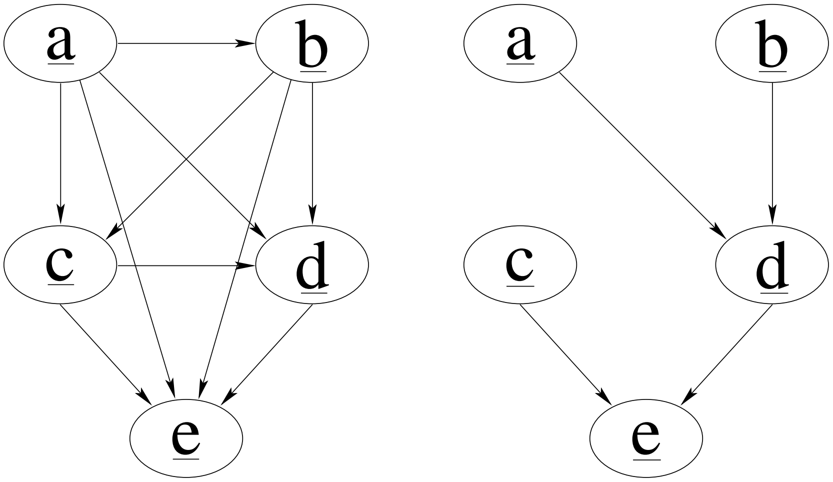


Figure 5.1: Two Bayesian network structures. The left side shows the structure for a domain where no simplification could be made, and thus has a fully connected structure. The graph on the right shows the structure where the following conditional independencies hold: $(\,\underline{b} \perp \underline{a}\,|\,\phi\,)$, $(\,\underline{c} \perp \underline{a},\underline{b}\,|\,\phi\,)$, $(\,\underline{d} \perp \underline{c}\,|\,\underline{a},\underline{b}\,)$ and $(\,\underline{e} \perp \underline{a},\underline{b}\,|\,\underline{c},\underline{d}\,)$. This gives rise to the following simplification of the joint distribution: $\mathrm{p}(\underline{a},\underline{b},\underline{c},\underline{d},\underline{e}) = \mathrm{p}(\underline{a})\,\mathrm{p}(\underline{b})\,\mathrm{p}(\underline{c})\,\mathrm{p}(\,\underline{d}\,|\,\underline{a},\underline{b}\,)\,\mathrm{p}(\,\underline{e}\,|\,\underline{c},\underline{d}\,)$.

One can choose only those simplifications that are consistent with the initial node ordering that has been chosen while applying the chain rule of probability. Conditional independence statements that are not consistent with this ordering cannot be expressed using a Bayesian network with that specific node ordering. As such, it is not possible to express the conditional independence statement $(\,\underline{b} \perp \underline{c}\,|\,\underline{d},\underline{e}\,)$ if the node ordering $\underline{a}$, $\underline{b}$, $\underline{c}$, $\underline{d}$ and $\underline{e}$ is used. By using the node ordering $\underline{c}$, $\underline{d}$, $\underline{e}$, $\underline{b}$, $\underline{a}$, the conditional independence statement at hand can be expressed.

Given an arbitrary set of conditional independence conditions, it is sometimes not possible to find a node ordering such that each conditional independence statement with a single variable on the left hand side, can be used for simplifying the factors. Conditional independencies with two or more variables on the left-hand side can similarly not always be represented.

Despite all this, a Bayesian network can still express a lot of different conditional independence statements, and the above restrictions are often not a problem in practice. Once a Bayesian network structure is given, this structure can be used to check if some conditional independence statement holds for that network structure. This can be computed using a graphical algorithm called *d-separation* [68].

The above states that not all conditional independence statements can be captured by a Bayesian network. On the other side, two different Bayesian network structures can sometimes express *exactly the same* conditional independence statements, making them equivalent. As a consequence, there is no difference between two equivalent Bayesian network structures from the probability distribution point of view; they can both represent the same set of distributions. It turns out that equivalent Bayesian network structures have the same skeleton, i.e., the same undirected structure. The difference between two equivalent network structures lies thus in the directionality of the edges. Chickering [17] derives a practical method to find out which arcs can be reversed without altering the conditional independence statements of a network structure. Figure 5.2 shows the general structure when an arc can be reversed. The nodes $\underline{a}$ and $\underline{b}$ in the middle must have the same parents if the arc between these two nodes has to be reversed without altering the conditional independence statements. We will not care too much about equivalent Bayesian network structures, as they will not matter to us.

Although most people who construct a Bayesian network structure by hand, like to think in terms of causal dependencies, it is not necessary to do so. It is convenient to draw an arc from $\underline{x}_1$ to $\underline{x}_2$ if $\underline{x}_1$ causes $\underline{x}_2$, but the two-node model with the reverse arc can express exactly the same distributions. If a Bayesian network structure is learned from a statistical data set, the resulting structure will not per se reflect the causal dependencies. The structure will be a reflection of the conditional independence statements that can be stated without compromising much of the capability of the network to model the data. In reality this will reflect only more or less the causal relations between the domain variables. The directionality of an arc can always be wrong and confounding factors prevent causal relationship inference in general observational studies like the ovarian tumour study introduced in Chapter 2.

A common, but simplistic network structure that is often used, is the *naive Bayes* structure (see Figure 5.3), and corresponds to the joint probability factorization

$$\mathrm{p}(\underline{a}, \underline{b}, \underline{c}, \underline{d}) = \mathrm{p}(\underline{d})\, \mathrm{p}(\, \underline{a} \,|\, \underline{d} \,)\, \mathrm{p}(\, \underline{b} \,|\, \underline{d} \,)\, \mathrm{p}(\, \underline{c} \,|\, \underline{d} \,). \tag{5.3}$$

This network structure is especially suitable for diagnosis problems. The variable $\underline{d}$ represents the diagnosis, while $\underline{a} \ldots \underline{c}$ are the symptoms. Specifying this

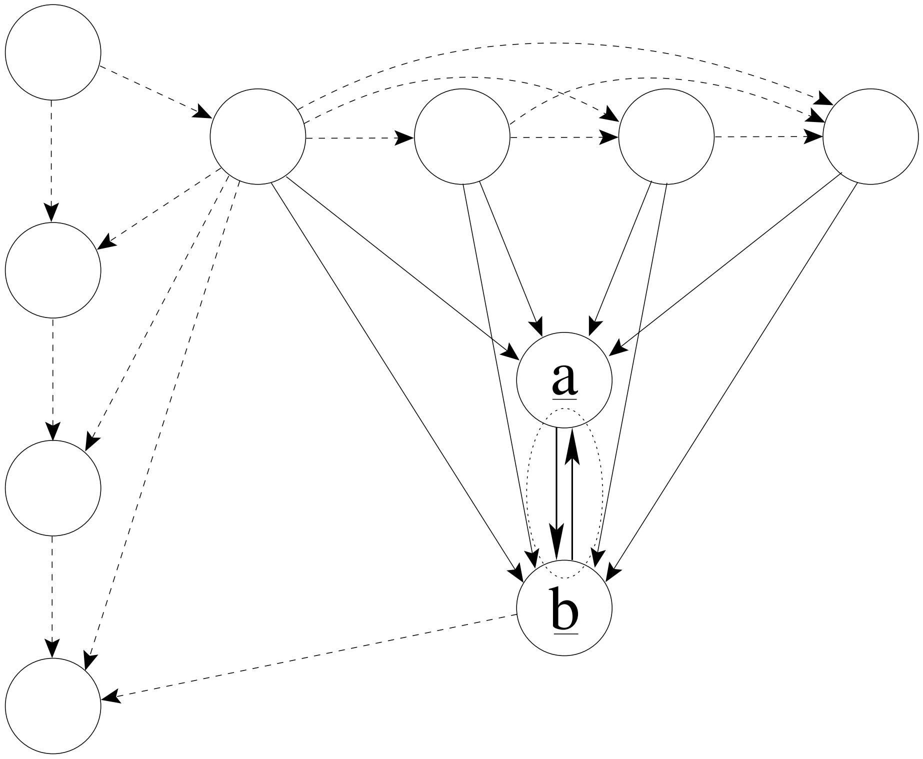


Figure 5.2: The general structure when an arc in a Bayesian network structure can be reversed. The nodes $\underline{a}$ and $\underline{b}$ in the middle must have to same parents if the arc between these two nodes has to be reversed without altering the conditional independence statements.

model is fairly easy: one only has to give the distribution of the diagnosis without any observation of the symptoms (neither the observation that a symptom is present nor absent), and the distribution of the symptoms if the diagnosis is known. Predicting the diagnosis when some of the symptoms are observed, is basically applying Bayes' rule; for example,

$$\begin{aligned} \mathrm{p}(\,\underline{d}\,|\,\underline{a},\underline{b},\underline{c}\,) &= \frac{\mathrm{p}(\,\underline{a},\underline{b},\underline{c}\,|\,\underline{d}\,)\,\mathrm{p}(\underline{d})}{\mathrm{p}(\underline{a},\underline{b},\underline{c})} \\ &= \frac{\mathrm{p}(\,\underline{a}\,|\,\underline{d}\,)\,\mathrm{p}(\,\underline{b}\,|\,\underline{d}\,)\,\mathrm{p}(\,\underline{c}\,|\,\underline{d}\,)\,\mathrm{p}(\underline{d})}{\sum_d \mathrm{p}(\,\underline{a}\,|\,\underline{d}=d\,)\,\mathrm{p}(\,\underline{b}\,|\,\underline{d}=d\,)\,\mathrm{p}(\,\underline{c}\,|\,\underline{d}=d\,)\,\mathrm{p}(\underline{d}=d)}. \end{aligned}$$

Making predictions for a general network structure will not be so straightforward, as is discussed in Section 5.2.4. This simplicity makes the naive Bayes network structure an often used structure. On the other hand, this model makes pretty strong assumptions on the domain to be applicable; it is assumed that the distributions of the symptoms — the child nodes — are independent if the

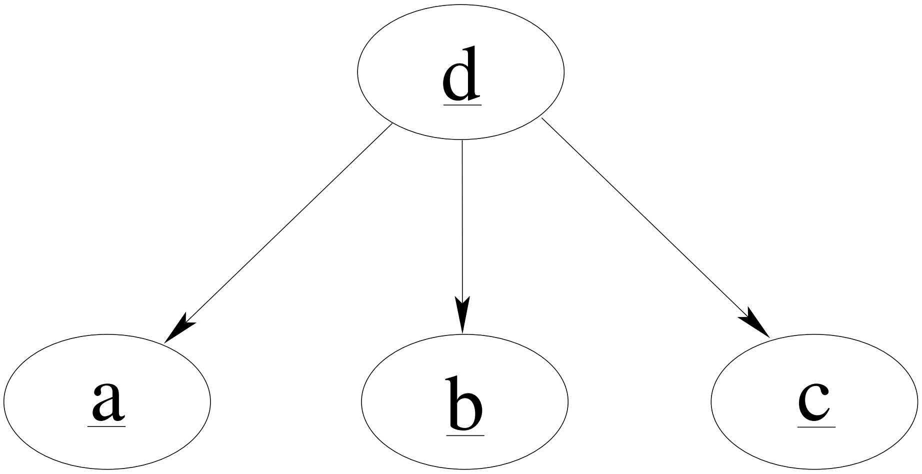


Figure 5.3: The naive Bayes network structure.

diagnosis $\underline{d}$ is observed.

$$\begin{aligned}
\mathrm{p}(\,\underline{a},\underline{b},\underline{c}\,|\,\underline{d}\,) &= \frac{\mathrm{p}(\underline{a},\underline{b},\underline{c},\underline{d})}{\mathrm{p}(\underline{d})} \\
&= \frac{\mathrm{p}(\underline{d})\,\mathrm{p}(\,\underline{a}\,|\,\underline{d}\,)\,\mathrm{p}(\,\underline{b}\,|\,\underline{d}\,)\,\mathrm{p}(\,\underline{c}\,|\,\underline{d}\,)}{\mathrm{p}(\underline{d})} \\
&= \mathrm{p}(\,\underline{a}\,|\,\underline{d}\,)\,\mathrm{p}(\,\underline{b}\,|\,\underline{d}\,)\,\mathrm{p}(\,\underline{c}\,|\,\underline{d}\,).
\end{aligned} \tag{5.4}$$

We used the factorization of the naive Bayes model (Equation 5.3) in Equation 5.4. If the diagnosis is not known, the symptoms are dependent on each other through the unknown $\underline{d}$.

For the opposite structure, displayed on Figure 5.4 and corresponding to the factorization

$$\mathrm{p}(\underline{a},\underline{b},\underline{c},\underline{d}) = \mathrm{p}(\underline{a})\,\mathrm{p}(\underline{b})\,\mathrm{p}(\underline{c})\,\mathrm{p}(\,\underline{d}\,|\,\underline{a},\underline{b},\underline{c}\,), \tag{5.5}$$

$\underline{a}$, $\underline{b}$ and $\underline{c}$ are independent if $\underline{d}$ is not known:

$$\begin{aligned}
\mathrm{p}(\underline{a},\underline{b},\underline{c}) &= \sum_d \mathrm{p}(\underline{a},\underline{b},\underline{c},\underline{d}=d) \\
&= \sum_d \mathrm{p}(\underline{a})\,\mathrm{p}(\underline{b})\,\mathrm{p}(\underline{c})\,\mathrm{p}(\,\underline{d}=d\,|\,\underline{a},\underline{b},\underline{c}\,) \\
&= \mathrm{p}(\underline{a})\,\mathrm{p}(\underline{b})\,\mathrm{p}(\underline{c})\sum_d \mathrm{p}(\,\underline{d}=d\,|\,\underline{a},\underline{b},\underline{c}\,) \\
&= \mathrm{p}(\underline{a})\,\mathrm{p}(\underline{b})\,\mathrm{p}(\underline{c}).
\end{aligned} \tag{5.6}$$

Again, we used the factorization of the joint distribution (Equation 5.5) in Equation 5.6. Observing $\underline{d}$ makes them dependent on each other.

Specifying such a reversed naive Bayes model in the discrete case is much harder, as one has to specify the distribution $\mathrm{p}(\,\underline{d}\,|\,\underline{a},\underline{b},\underline{c}\,)$ of the node $\underline{d}$ for each different possible value configuration of the parents. This results in specifying $\mathcal{O}(d^k)$ parameters (the number of different value combinations for the parents) instead of $\mathcal{O}(k \cdot d)$ in the naive Bayes' case (for each of the d possible values of $\underline{d}$, we have to specify the distribution of the k child nodes). The number of parents (or children in the naive Bayes' case) is k and d represents the number of possible values for these nodes. The naive Bayes structure from Figure 5.3 therefore needs only $3d(d-1)$ parameters to specify the distribution of the children, while the reversed network structure from Figure 5.4 will need $d^3(d-1)$ parameters to specify the distribution of the child $\underline{d}$. Note that both models also need a few parameters to specify the distribution of the parents. For each parent, there are $d-1$ parameters needed when table distributions are used. The total number of parameters for the naive Bayes model is therefore $3d(d-1)+d-1$, while for the reversed naive Bayes model increases to $d^3(d-1)+3(d-1)$.

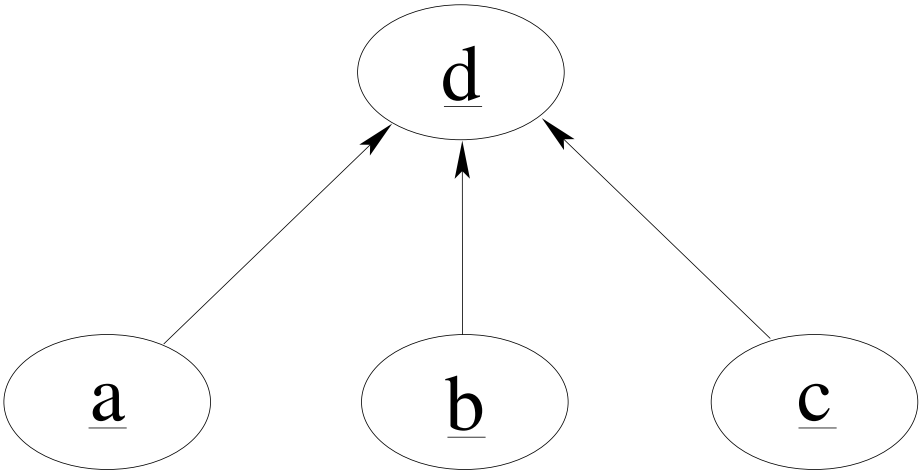


Figure 5.4: The reversed naive Bayes structure.

5.1.2 Local parameters and conditional dependency models

Besides the network structure, we also have to specify the *local dependency models* $\mathrm{p}(\,\underline{x}_i\,|\,\pi(\underline{x}_i)\,)$. They specify how the parents of a node influence the distribution of the child variable. We are free to choose any conditional distribution we like, but a few considerations should be made; if it is the purpose to perform inference with the model, the form of these conditional distributions is of crucial importance for the algorithms that need to be designed.

For modelling discrete distributions, it is common and practical to use a different table distribution for each variable and each different parental configuration. The computations for these conditional distributions can be carried out exactly, but the number of parameters involved grows exponentially with the number of parents.

For modelling continuous distributions, a regression-like distribution is often chosen; some function of the parental values specifies the mean of the conditional distribution. The distribution around this mean is often, but not necessarily, a Gaussian with fixed variance. When the model for the mean is a linear function for each variable, we are actually modelling a multivariate Gaussian distribution with a sparse inverse covariance matrix.

5.2 Algorithms

To specify a Bayesian network, one has to give the network structure and the conditional probability distributions that express the dependencies of the parents on its child. We denote the structure of a Bayesian network with $\mathcal{S}_{\mathrm{bn}}$, while the parameters are represented with $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. Depending on the domain at hand and the available information, we have to design the network structure and/or local dependency models by hand or infer them from a statistical data set. The bilateral structure of a Bayesian network is maintained in the learning process: we will first learn the structure and then learn its parameters.

Once a Bayesian network is specified, we are ready to use it. The usage can vary from generating random samples from the modelled distribution over computing specific marginal conditional distributions to sensitivity analysis, explaining why it takes certain decisions, or finding out which variable should be observed next to reduce most of the uncertainty of some conditional distribution.

5.2.1 Structure learning

There are several procedures to learn the structure of a Bayesian network or perform inference on the structure [30], all based on a different principle. Among these, we have maximum likelihood, Bayesian learning, minimum description length, Akaike information criterion, entropy-based learning, and so on. We will discuss Bayesian learning because it has a clear probabilistic interpretation. At first, we concentrate on the general setup and will discuss the details for some specific local dependency models in Section 5.3.

The purpose is to find that network structure that maximizes $\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}} \,|\, \mathbf{D}, \xi\,)$. Let us first look into this posterior probability of a network structure $\mathcal{S}_{\mathrm{bn}}$ given a data set, where each network structure has the same prior probability and the samples from the data set are independent once the model (both the structure and parameters) is known

$$\begin{aligned}
\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}}\,|\,\mathbf{D},\xi\,) &= \frac{\mathrm{p}(\,\mathbf{D}\,|\,\mathcal{S}_{\mathrm{bn}}\,)\,\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}}\,|\,\xi\,)}{\mathrm{p}(\mathbf{D})} \\
&\propto \mathrm{p}(\,\mathbf{D}\,|\,\mathcal{S}_{\mathrm{bn}}\,) \\
&= \int_{\mathbf{\Theta}} \mathrm{p}(\,\mathbf{D}\,|\,\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}\,)\,\mathrm{p}(\boldsymbol{\theta})\,d\boldsymbol{\theta} \\
&= \int_{\mathbf{\Theta}} \prod_{i=1}^{n} \mathrm{p}(\,\boldsymbol{x}^i\,|\,\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}\,)\,\mathrm{p}(\boldsymbol{\theta})\,d\boldsymbol{\theta} \\
&= \int_{\mathbf{\Theta}} \prod_{i=1}^{n}\prod_{j=1}^{v} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i),\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}\,)\,\mathrm{p}(\boldsymbol{\theta})\,d\boldsymbol{\theta} \\
&= \int_{\mathbf{\Theta}} \prod_{j=1}^{v}\prod_{i=1}^{n} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i),\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}_j\,)\,\mathrm{p}(\boldsymbol{\theta})\,d\boldsymbol{\theta}.
\end{aligned}$$

We treat the probability of the data $\mathrm{p}(\mathbf{D})$ as a constant and assume that we have no a priori knowledge about the structure. Therefore, $\mathrm{p}(\mathbf{D})$ and $\mathrm{p}(\mathcal{S}_{\mathrm{bn}})$ are nothing more than constant multiplication factors, which we drop to make the essence more clear. The integration domain is indicated with $\mathbf{\Theta}$, which is the set of all possible parametrizations.

We made the assumption that we can divide the parameters $\boldsymbol{\theta}$ into disjunct subsets $\boldsymbol{\theta}_j$, where $\boldsymbol{\theta}_j$ is sufficient to describe the distribution of $\underline{x}_j$ given its parents:

$$\mathrm{p}(\,\underline{x}_j\,|\,\pi(\underline{x}_j),\boldsymbol{\theta}\,) = \mathrm{p}(\,\underline{x}_j\,|\,\pi(\underline{x}_j),\boldsymbol{\theta}_j\,). \tag{5.7}$$

Our next and related assumption is called *global parameter independence*, and means that we can treat these parameters $\boldsymbol{\theta}_j$ corresponding to different local dependency models as being independent before we observe any data

$$\mathrm{p}(\boldsymbol{\theta}) = \prod_{j=1}^{v} \mathrm{p}(\boldsymbol{\theta}_j). \tag{5.8}$$

This allows us to continue to work out our expression for the a posteriori probability of the network structure given the data

$$\begin{aligned}
\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}}\,|\,\mathbf{D}\,) &\propto \int_{\mathbf{\Theta}} \prod_{j=1}^{v}\prod_{i=1}^{n} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i),\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}_j\,) \prod_{k=1}^{v} \mathrm{p}(\boldsymbol{\theta}_k)\,d\boldsymbol{\theta} \\
&= \int_{\mathbf{\Theta}} \prod_{j=1}^{v}\prod_{i=1}^{n} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i),\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}_j\,)\,\mathrm{p}(\boldsymbol{\theta}_j)\,d\boldsymbol{\theta} \\
&= \prod_{j=1}^{v} \int_{\mathbf{\Theta}_j} \prod_{i=1}^{n} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i),\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}_j\,)\,\mathrm{p}(\boldsymbol{\theta}_j)\,d\boldsymbol{\theta}_j \qquad (5.9) \\
&= \prod_{j=1}^{v} L(\,j\,|\,\pi_j\,).
\end{aligned}$$

This last equation indicates that the structure posterior decomposes into a product of factors where each factor contains information about only one child node $\underline{x}_j$ and its parents. We denote such a factor with $L(\, j \,|\, \pi_j \,)$. This important observation suggests that we can find the *maximum a posteriori* network structure by searching for an optimal parental set for each variable independently and combine these local optimizations into one structure. This procedure is correct if the order of the variables when applying the chain rule of probability is known.

Unfortunately, it is most of the time unclear how to choose the initial node ordering. It is then a common strategy to start with generating a set of random node orderings. For each of these node orderings, we will learn a network structure, and keep thát structure that had the highest probability among all the learned structures.

Within a given node ordering, we have to find the optimal parents for each node. When searching the parents for node $\underline{x}_j$, we can choose only nodes from the set $\{\underline{x}_1, \ldots, \underline{x}_{j-1}\}$ as potential parents to ensure that the result will be a directed acyclic graph. Only for very small values of j, we can evaluate $L(\, j \,|\, \pi_j \,)$ for each possible set of parents, as the total number of possible subsets increases very fast with the number of possible elements we can choose from. Although we could restrict us to look only for that set of parents consisting only of a few nodes, this would restrict the Bayesian network space we are considering seriously. Therefore we will extend this *exhaustive* search with a *greedy-based* search. Instead of looking at all the possible sets, we try to extend the current parent set with only *one* variable at a time. We add thát node — if any — that increases $L(\, j \,|\, \pi_j \,)$ maximally and try to add another node in the following iteration:

```
1. $StructureProbability \leftarrow 0$.
2. $CurrentDAG \leftarrow \emptyset$.
3. Repeat for the number of node permutations that should be tried:
   (a) Choose a random node permutation $(\underline{\tilde{x}}_1, \ldots, \underline{\tilde{x}}_v)$ such that $\{\underline{\tilde{x}}_1, \ldots, \underline{\tilde{x}}_v\} = \{\underline{x}_1, \ldots, \underline{x}_v\}$.
   (b) For $j = 1$ to $v$ do
       i. Find $\pi_j$ of size $M$ or smaller that maximizes $L(\, j \,|\, \pi_j \,)$ using an exhaustive search method.
       ii. For each $\underline{x}^* \in \{\underline{\tilde{x}}_1, \ldots, \underline{\tilde{x}}_{j-1}\} \setminus \pi_j$, evaluate $L(\, j \,|\, \pi_j \cup \underline{x}^* \,)$.
       iii. If $\max_{\underline{x}^*} L(\, j \,|\, \pi_j \cup \underline{x}^* \,) > L(\, j \,|\, \pi_j \,)$, set $\pi_j \leftarrow \pi_j \cup \mathrm{argmax}_{\underline{x}^*}(L(\, j \,|\, \pi_j \cup \underline{x}^* \,))$ and go to (ii).
   (c) If $StructureProbability < \prod_j L(\, j \,|\, \pi_j \,)$, set $StructureProbability \leftarrow \prod_j L(\, j \,|\, \pi_j \,)$ and $CurrentDAG \leftarrow (j \leftarrow \pi_j)_{j=1}^{v}$.
4. Return $CurrentDAG$.
```

In the above algorithm, *CurrentDAG* represents the current optimal directed acyclic graph we have already found and is represented with the set of edges connecting each node with its parents. This is written down using the notation $(j \leftarrow \pi_j)_{j=1}^{v}$.

Instead of learning the structure for a large number of initial node permutations, we developed an algorithm that searches for a good node ordering. We denote with A the set of nodes that are already ordered satisfactorily and with B the set of nodes that are not ordered yet. The following procedure will gradually fill A and hereby generating a usable ordering and constructing the optimal network structure with respect to this given ordering:

1. Set $A \leftarrow \emptyset$ and $B \leftarrow \{\underline{x}_1, \ldots, \underline{x}_v\}$.

2. Choose $\underline{x}^*$ such that $\prod_{y \in B \setminus \underline{x}^*} \max_{\pi_y \subset A \cup \underline{x}^*} L(\, y \,|\, \pi_y \,)$ is maximal.

3. Learn the optimal parents for $\underline{x}^*$ with step (b) from the above greedy hill climbing procedure where we can choose parents from A.

4. Set $A \leftarrow A \cup \underline{x}^*$ and goto 1 if $B \neq \emptyset$.

Step 2 in the above procedure selects that node from B that helps predicting the other variables in B the most with the additional help of (some) variables from A.

No matter which of the two procedures is used, the local substructure probability $L(\, j \,|\, \pi_j \,)$ is the basic element and has to be computed very often. As usual, it is more convenient to compute the logarithm of this probability, transforming the products into summations. It also pays off to cache these individual values to avoid repeatedly computing the same local substructure probabilities.

We discussed a procedure that tries to approximate the maximum a posteriori network structure. We work in the Bayesian framework by integrating over the parameters $\boldsymbol{\theta}$. This anticipates the overtraining that occurs when the maximum likelihood procedure

$$\mathcal{S}_{\mathrm{bn}} = \mathrm{argmax}_{\boldsymbol{\theta}, \mathcal{S}_{\mathrm{bn}}} (\mathrm{p}(\, \mathbf{D} \,|\, \boldsymbol{\theta}, \mathcal{S}_{\mathrm{bn}} \,))\big|_{\mathcal{S}_{\mathrm{bn}}}$$

is applied. In the above formula, $\big|_{\mathcal{S}_{\mathrm{bn}}}$ indicates the restriction of the result $(\boldsymbol{\theta}^*, \mathcal{S}_{\mathrm{bn}}^*)$ from the argmax procedure to only the network structure $\mathcal{S}_{\mathrm{bn}}^*$.

Although we should sum over all network structures and weight each term with the corresponding structure probability to perform pure Bayesian inference, we will be satisfied and continue with only our (approximation of the) maximum a posteriori network structure.

Procedures that learn the structure of a Bayesian network in the presence of missing values or hidden variables do exist and can be found in [28]. Such procedures use a Gibbs sampling or EM approach to find the solution in an iterative manner.

5.2.2 Parameter learning

Once the structure is fixed, we can start learning the parameters of the conditional probability distributions that describe the local dependency models. By learning, we can mean finding the maximum likelihood parametrizations, the maximum a posteriori parametrizations or the full Bayesian posterior parameter distribution.

Finding the maximum likelihood parameters decomposes into finding the parameters for each local dependency model under the assumption stated in Equation 5.7

$$\begin{aligned}\mathcal{L}(\,\boldsymbol{\theta}\,|\,\mathbf{D}\,) &= \mathrm{p}(\,\mathbf{D}\,|\,\boldsymbol{\theta}, \mathcal{S}_{\mathrm{bn}}\,)\\ &= \prod_{i=1}^{n} \mathrm{p}(\,\boldsymbol{x}^i\,|\,\boldsymbol{\theta}, \mathcal{S}_{\mathrm{bn}}\,)\\ &= \prod_{i=1}^{n}\prod_{j=1}^{v} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i), \boldsymbol{\theta}_j, \mathcal{S}_{\mathrm{bn}}\,)\\ &= \prod_{j=1}^{v}\left(\prod_{i=1}^{n} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i), \boldsymbol{\theta}_j, \mathcal{S}_{\mathrm{bn}}\,)\right).\end{aligned}$$

This maximum likelihood setup has only a good behaviour when many samples are present. For the table distribution case, as discussed in Section 5.3.1, this imposes strong assumptions on the data set and even then this procedure results in unreasonable parameter settings when only a small number of samples are present. Therefore, we look for the maximum a posteriori parametrization. Using the decomposable prior assumption (Equation 5.8), finding the optimal parameters decomposes into finding the parameters for each local substructure separately

$$\begin{aligned}\mathrm{p}(\,\boldsymbol{\theta}\,|\,\mathcal{S}_{\mathrm{bn}}, \mathbf{D}\,) &= \frac{\mathrm{p}(\,\mathbf{D}\,|\,\boldsymbol{\theta}, \mathcal{S}_{\mathrm{bn}}\,)\,\mathrm{p}(\,\boldsymbol{\theta}\,|\,\mathcal{S}_{\mathrm{bn}}\,)}{\mathrm{p}(\,\mathbf{D}\,|\,\mathcal{S}_{\mathrm{bn}}\,)}\\ &\propto \prod_{j=1}^{v}\left(\prod_{i=1}^{n} \mathrm{p}(\,x_j^i\,|\,\pi(x_j^i), \boldsymbol{\theta}_j, \mathcal{S}_{\mathrm{bn}}\,)\,\mathrm{p}(\,\boldsymbol{\theta}_j\,|\,\mathcal{S}_{\mathrm{bn}}\,)\right).\end{aligned}$$

In some cases, we can even derive an analytic formulation for this posterior parameter distribution, so there is then no reason to work with only the maximum a posteriori parametrization. Dasgupta [22] describes the sample complexity of learning the parameters of a Bayesian network.

5.2.3 Generating random vectors

As indicated before, a Bayesian network is nothing more than a sparse representation of a joint distribution. Once both the structure and the parameters are learned, the chain rule formulation of this joint distribution — Equation 5.1 — can be immediately used to generate random vectors efficiently from this joint distribution if we can generate samples from the individual conditional distributions:

Make sure that $\pi(\underline{x}_i) \subset \{\underline{x}_1, \ldots, \underline{x}_{i-1}\}$, $\forall i$.

1. Generate x_1 according to $\mathrm{p}(\underline{x}_1)$.
2. Generate x_2 according to $\mathrm{p}(\,\underline{x}_2 \,|\, \pi(\underline{x}_2)\,)$.
3. ...

v. Generate x_v according to $\mathrm{p}(\,\underline{x}_v \,|\, \pi(\underline{x}_v)\,)$.

Return $(x_1, x_2, \ldots, x_v)$.

Vectors generated with the above procedure will be independent of each other.

5.2.4 Inference

In addition to generating samples from the joint distribution, we can also use our Bayesian network to answer questions. With a question, we simply mean that we want to compute some specific conditional distribution, something we are interested in. For the ovarian tumour case, this could be

> What is the probability that a tumour is malignant when the patient is 66 years old and has 3 children?

This distribution can then be used to make a prediction or classification.

Computing a general univariate conditional distribution can be done with the *probability propagation in tree of cliques* algorithm [45]. This is a quite complicated algorithm that involves some graphical steps (moralizing and triangulating the graph to obtain a tree of cliques) that are followed by inserting and propagating the evidence (the condition) into the tree of cliques. The result of this procedure is a conditional distribution over the nodes contained in each clique, conditioned on the evidence. From this joint distribution over the clique variables, the desired univariate conditional distribution can be computed using marginalization.

For this procedure to work, it is necessary that such a joint distribution over a set of variables can be stored and that a conditional distribution can be multiplied into this. For table or linear-Gaussian dependency models, this is the case. If other distributions are used, one can try to approximate it using the techniques explained in [55].

Unfortunately, the probability propagation in tree of cliques procedure is proven to be NP hard [18], which means that computing conditional distributions from a large densely connected network becomes intractable. Variational approximations exist which approximate the inference in the intractable network with inference performed in a tractable network with minimal Kullback-Leibler divergence [51, 88].

Computing the distribution of more than one variable is computed by applying the chain rule of probability and using the above technique several times.

5.2.5 Graph layout

A last operation on Bayesian networks that we will discuss is about interpretation and human interface, instead of the modelling or usage of the model itself. Even for only a few nodes, looking at and interpreting a graph is influenced for a large portion by its 2D graphical layout. An optimal layout would be one where the nodes are spread out uniformly and a minimal number of edge crossings occur. Finding this optimal layout can be difficult, but good approximate techniques exist, for instance based on simulating a physical system where each edge is replaced by a spring.

Instead of implementing and using one of these algorithms, we tried and tested a novel approach based on a self-organizing map. Our demands are not that severe, so we do not need maximal performance, and it seemed a nice idea to test.

The self-organising map concept [53, 54] was originally developed by Kohonen to create a two-dimensional map of a higher dimensional data set. Usually, a two-dimensional regular lattice structure is used to approximate the higher dimensional data set. This approximation means that the algorithm will try to learn the data distribution by placing the nodes of the self-organizing map in the vicinity of the data points. Because the nodes of the map are connected through the lattice structure, moving one node in a certain direction, will also move those nodes that are connected via a direct link by some amount, the nodes that are connected with two edges a smaller amount etc. The net effect is that the nodes of the map will try to approximate the data distribution with the constraint of the network structure.

Our approach is to create the data set ourselves, namely a two-dimensional uniformly distributed data set in some window. Instead of using a regular lattice map, we use the network we want to lay out. The self-organising map algorithm will now try to approximate the data distribution under the constraints of the network structure. This means that the algorithm will try to spread the network nodes as uniformly as possible, under the constraint of the network structure. This is exactly the behaviour we expect from a graph layout algorithm and the performance was more than suitable. An example of this procedure is presented in Figure 5.5 and Figure 5.6. The first network has random node coordinates, while the latter has coordinates computed with our self-organising map technique.

5.3 Different types of networks

The type of the conditional distributions that is chosen to describe the local dependency models, is of crucial importance. Depending on the variable types (discrete, continuous, or mixed), other distributions will be considered. In addition to these local dependencies, it is important to select a distribution and its corresponding parameters to result in efficient computations. For structure learning, we should be able to compute the local substructure probabilities

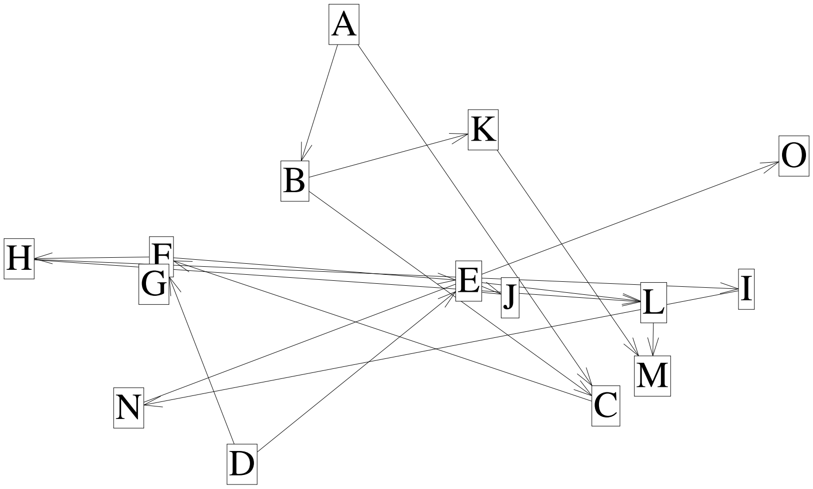


Figure 5.5: A network with a random layout.

$L(\, j \,|\, \pi_j \,)$ efficiently. Parameter learning is also influenced by the choice of the distribution and the prior. Finally, performing inference has other specific demands that can influence the choice of the distributions.

We discuss a few commonly used alternatives.

5.3.1 Table distributed variables

A widely used type of dependency model is based on the table distribution (see Section 3.1.3) and is applicable only for discrete variables. This choice results in nice closed form expressions for the local substructure probabilities and the posterior parameter distribution together with exact procedures for performing inference. The table distribution can describe *any* discrete distribution On the downside, there is a lot of parameters involved, which can result in slow learning and computational problems.

Conditional distribution

To describe the local dependency models, we use a *separate* table distribution for each variable *and* parental configuration [19, 41]. We denote the parameters describing the conditional table distribution for variable $\underline{x}_j$ with $\boldsymbol{\theta}_j$, which is a collection of the table parameters $\boldsymbol{\theta}_{j,\pi_k}$ for each different parental configuration π_k:

$$\mathrm{p}(\, \underline{x}_j = m \,|\, \pi(\underline{x}_j) = \pi_k, \boldsymbol{\theta}_j \,) = \boldsymbol{\theta}^{m}_{j,\pi_k}.$$

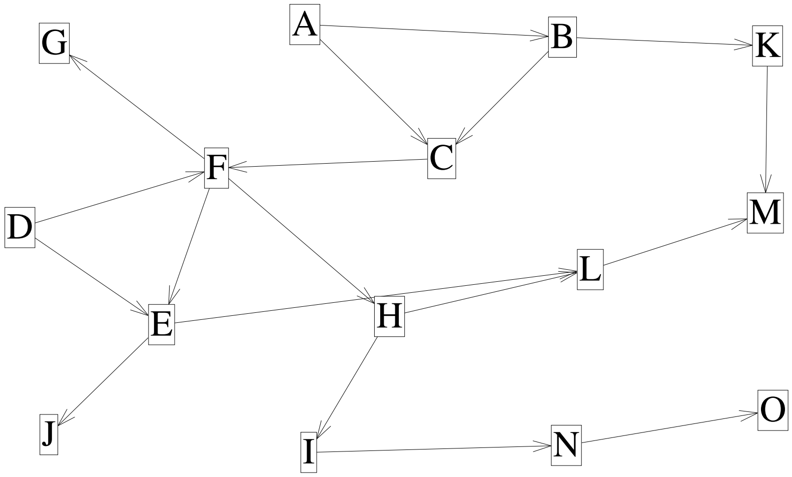


Figure 5.6: The same network from Figure 5.5, with a layout computed using the self-organising map approach.

This conditional distribution can represent any conditional discrete distribution. The number of parameters involved increases exponentially with the number of parents. If all the nodes can have only two possible values, we still need $2^{\#\pi_j}$ parameters to store this distribution, where $\#\pi_j$ is the number of parents for node j. Even if storing these parameters is no problem, estimating them often requires many data records. This problem can be dealt with by using techniques explained in [31].

Prior for the parameters

Heckerman [41] mentions that the only sensible choice for the parameter prior is based on the Dirichlet distribution:

$$\begin{aligned}
\mathrm{p}(\,\boldsymbol{\theta}_{j,\pi_k}\,|\,m_1,\ldots,m_d\,) &= \frac{\Gamma(\sum_l m_l)}{\prod_l \Gamma(m_l)}\prod_l (\boldsymbol{\theta}^l_{j,\pi_k})^{m_l-1} \quad \text{with} \\
\sum_l \boldsymbol{\theta}^l_{j,\pi_k} &= 1 \\
\mathrm{E}[\boldsymbol{\theta}_{j,\pi_k}] &= \left(\frac{m_1}{\sum_l m_l},\frac{m_2}{\sum_l m_l},\ldots,\frac{m_d}{\sum_l m_l}\right) \\
\mathrm{V}[\boldsymbol{\theta}^l_{j,\pi_k}] &= \frac{m_l(1-m_l/\sum_r m_r)}{(\sum_r m_r+1)\sum_r m_r}.
\end{aligned}$$

The *hyper-parameters* $(m_1,\ldots,m_d)$ are also called pseudo-counts, as they have the direct interpretation of the equivalent number of samples that the prior is worth. This makes it possible to specify an informative prior simply by giving an estimate of the number of times a certain combination of the random variables has been observed a priori. Note that, if we choose the hyperparameters to be 1, we have the uniform distribution on the set of positive numbers that sum to 1, which corresponds to saying that we saw each possible child-parent combination one time.

The Dirichlet distribution suits very well to describe the distribution of the table parameters corresponding to the distribution of a certain variable $\underline{x}_j$ given a fixed parental configuration $\pi(\underline{x}_j) = \pi_k$. Since we chose a different table distribution for each different parental configuration, it is natural to choose a product of independent Dirichlet distributions to describe the prior distribution of *all* the parameters concerning a certain variable $\underline{x}_j$. This assumption is called *local parameters independence*:

$$\begin{aligned} \mathrm{p}(\boldsymbol{\theta}_j) \quad &= \quad \prod_{k=1}^{q_j} \mathrm{p}(\, \boldsymbol{\theta}_{j,\pi_k} \,|\, m_1, \ldots, m_d \,) \text{ with} \qquad (5.10) \\ & \qquad \mathrm{p}(\, \boldsymbol{\theta}_{j,\pi_k} \,|\, m_1, \ldots, m_d \,) \text{ Dirichlet and} \\ & \qquad \boldsymbol{\theta}_j = (\boldsymbol{\theta}_{j,\pi_1}, \ldots, \boldsymbol{\theta}_{j,\pi_{q_j}}). \end{aligned}$$

We denote the total number of different parental configurations for $\underline{x}_j$ with q_j.

Posterior for the parameters

For the discrete table variable case without missing values, finding the maximum likelihood parametrization for $\boldsymbol{\theta}^l_{j,\pi_k}$ is a simple matter of counting the frequency that $\underline{x}_j = l$ within the subset of data vectors that are compatible with the parental configuration π_k. If we denote with q_j the number of different parental configurations for node j, the expression for the likelihood becomes

$$\begin{aligned} \mathrm{p}(\, \mathbf{D} \,|\, \boldsymbol{\theta}, \mathcal{S}_{\mathrm{bn}} \,) \quad &= \quad \prod_{i=1}^{n} \prod_{j=1}^{v} \mathrm{p}(\, x^i_j \,|\, \pi(x^i_j), \boldsymbol{\theta}, \mathcal{S}_{\mathrm{bn}} \,) \\ &= \quad \prod_{j=1}^{v} \prod_{i=1}^{n} \boldsymbol{\theta}^{x^i_j}_{j,\pi(x^i_j)} \\ &= \quad \prod_{j=1}^{v} \prod_{k=1}^{q_j} \prod_{l=1}^{d} (\boldsymbol{\theta}^l_{j,\pi_k})^{n_{jkl}} \quad \text{with} \\ \sum_{l=1}^{d} \boldsymbol{\theta}^l_{j,\pi_k} \quad &= \quad 1 \quad \text{and} \quad \boldsymbol{\theta}^l_{j,\pi_k} \geq 0. \end{aligned}$$

This expression has its maximum for the expected counting frequency

$$\boldsymbol{\theta}^l_{j,\pi_k} = \frac{n_{jkl}}{\sum_{r=1}^{d} n_{jkr}},$$

where n_{jkl} is the number of samples in the data set where $\underline{x}_j = l$ while its parents are in the k^{th} parental configuration $\pi(\underline{x}_j) = \pi_k$.

Finding the maximum a posteriori parametrization is very similar:

$$\begin{aligned}
\mathrm{p}(\boldsymbol{\theta} \,|\, \mathbf{D}, \mathcal{S}_{\mathrm{bn}}) &\propto \mathcal{L}(\boldsymbol{\theta} \,|\, \mathbf{D}) \,\mathrm{p}(\boldsymbol{\theta}) \\
&= \prod_{j=1}^{v} \prod_{i=1}^{n} \boldsymbol{\theta}_{j,\pi(x_j^i)}^{x_j^i} \mathrm{p}(\boldsymbol{\theta}_j) \\
&= \prod_{j=1}^{v} \prod_{k=1}^{q_j} \prod_{l=1}^{d} (\boldsymbol{\theta}_{j,\pi_k}^{l})^{n_{jkl}} \mathrm{p}(\boldsymbol{\theta}_{j,\pi_k}^{l}) \\
&= \prod_{j=1}^{v} \prod_{k=1}^{q_j} \prod_{l=1}^{d} (\boldsymbol{\theta}_{j,\pi_k}^{l})^{n_{jkl}+m_{jkl}-1}, \qquad (5.11)
\end{aligned}$$

where m_{jkl} denotes the Dirichlet hyper-parameter that corresponds to the event that $\underline{x}_j = l$ while the parents of node j are $\pi(\underline{x}_j) = \pi_k$.

Equation 5.11 tells us that the local parameter independence is also valid for the a posteriori distribution if it was valid for the a priori distribution, as denoted by Equation 5.10.

By a completely similar reasoning as for the likelihood case, we come to the following formulation for the maximum a posteriori parametrization:

$$\boldsymbol{\theta}_{j,\pi_k}^{l} = \frac{n_{jkl} + m_{jkl} - 1}{\sum_{r=1}^{d} n_{jkr} + \sum_{r=1}^{d} m_{jkr} - d}.$$

We have a closed form formula for the maximum a posteriori parametrization and Equation 5.11 even tells us that we can specify the posterior distribution exactly; this posterior stays a Dirichlet distribution, we only have to update the prior hyper-parameters m_{jkl} with the data counts n_{jkl} by a simple addition.

Probability of local substructure

We can write out the posterior structure probability like

$$\begin{aligned}
\mathrm{p}(\mathcal{S}_{\mathrm{bn}} \,|\, \mathbf{D}) &\propto \prod_{j=1}^{v} L(j \,|\, \pi_j) \\
&= \prod_{j=1}^{v} \int_{\boldsymbol{\Theta}_j} \prod_{i=1}^{n} \mathrm{p}(x_j^i \,|\, \pi(x_j^i), \mathcal{S}_{\mathrm{bn}}, \boldsymbol{\theta}_j) \,\mathrm{p}(\boldsymbol{\theta}_j) \, d\boldsymbol{\theta}_j \\
&= \prod_{j=1}^{v} \int_{\boldsymbol{\Theta}_j} \prod_{i=1}^{n} \mathrm{p}(x_j^i \,|\, \pi(x_j^i), \mathcal{S}_{\mathrm{bn}}, \boldsymbol{\theta}_j) \prod_{k=1}^{q_j} \mathrm{p}(\boldsymbol{\theta}_{j,\pi_k}) \, d\boldsymbol{\theta}_j \qquad (5.12) \\
&= \prod_{j=1}^{v} \int_{\boldsymbol{\Theta}_j} \prod_{k=1}^{q_j} \frac{\Gamma(\sum_l m_{jkl})}{\prod_l \Gamma(m_{jkl})} \prod_{l=1}^{d} (\boldsymbol{\theta}_{j,\pi_k}^{l})^{n_{jkl}+m_{jkl}-1} d\boldsymbol{\theta}_j \\
&= \prod_{j=1}^{v} \prod_{k=1}^{q_j} \frac{\Gamma(\sum_l m_{jkl})}{\prod_l \Gamma(m_{jkl})} \frac{\prod_l \Gamma(n_{jkl} + m_{jkl})}{\Gamma(\sum_l n_{jkl} + \sum_l m_{jkl})}.
\end{aligned}$$

In Equation 5.12, we used the local parameter independence assumption of Equation 5.10 and used the same reasoning as in Equation 5.11 to continue.

This gives us a closed form formula for $L(\, j \,|\, \pi_j \,)$:

$$L(\, j \,|\, \pi_j \,) = \prod_{k=1}^{q_j} \frac{\Gamma(\sum_l m_{jkl})}{\prod_l \Gamma(m_{jkl})} \frac{\prod_l \Gamma(n_{jkl} + m_{jkl})}{\Gamma(\sum_l n_{jkl} + \sum_l m_{jkl})}.$$

If we choose the prior counts independent from the parents or the *number* of parents, we can drop the term

$$\frac{\Gamma(\sum_l m_{jkl})}{\prod_l \Gamma(m_{jkl})}$$

from the above expression.

5.3.2 Linear-Gaussian distributed variables

Another widely used class of conditional dependency models, this time exclusively for continuous variables, are the linear-Gaussian conditional distributions. These types of networks are nothing more or less than a multivariate Gaussian distribution. The network structure will result in a sparse inverse of the covariance matrix.

Conditional distribution

In a linear-Gaussian dependency model, the child node $\underline{x}_j$ has a Gaussian distribution with a fixed conditional variance σ_j^2 and a mean that depends in a linear way on the values π_{ji} of the parents

$$\begin{aligned} \mathrm{p}(\, x_j \,|\, \pi_j \,) &\sim \mathcal{N}(\alpha_0 + \alpha_1 \pi_{j1} + \cdots + \alpha_p \pi_{jp}, \sigma_j^2) \\ &\sim \mathcal{N}(\mu(\pi_j), \sigma_j^2) \\ &\sim \mathcal{N}(\beta_0 + \sum_{i=1}^{p} \alpha_i (\pi_{ji} - \boldsymbol{\mu}_{\pi_i}), \sigma_j^2). \end{aligned} \tag{5.13}$$

We can either express the conditional mean using the parent values directly, or use their deviation from their mean.

The joint distribution described by such a Bayesian network is a multivariate Gaussian distribution. The opposite also holds, which means that each multivariate Gaussian distribution can be written down as a Bayesian network with linear-Gaussian dependency models. We will show this for only one variable $\underline{x}$

and its parents $\pi(\underline{x})$:

$$\begin{aligned}
\mathrm{p}(\pi(\underline{x}), \underline{x}) &\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
&= \frac{1}{2\pi^{(p+1)}/2|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}((\pi(\underline{x}),\underline{x})-(\boldsymbol{\mu}_\pi,\mu_x))\boldsymbol{\Sigma}^{-1}((\pi(\underline{x}),\underline{x})-(\boldsymbol{\mu}_\pi,\mu_x))^T} \\
&= \frac{1}{2\pi^{p/2}|\boldsymbol{\Sigma}_{\pi\pi}|} e^{-\frac{1}{2}(\pi(\underline{x})-\boldsymbol{\mu}_\pi)\boldsymbol{\Sigma}_{\pi\pi}^{-1}(\pi(\underline{x})-\boldsymbol{\mu}_\pi)} \times \\
&\quad \frac{1}{\sqrt{2\pi}|S_{xx}^{-1}|} e^{-\frac{1}{2}(\underline{x}-\mu_x+S_{xx}^{-1}\boldsymbol{S}_{x\pi}(\pi(\underline{x})-\boldsymbol{\mu}_\pi))S_{xx}(\underline{x}-\mu_x+S_{xx}^{-1}\boldsymbol{S}_{x\pi}(\pi(\underline{x})-\boldsymbol{\mu}_\pi))} \\
&= \mathcal{N}(\boldsymbol{\mu}_\pi, \boldsymbol{\Sigma}_{\pi\pi})\,\mathcal{N}(\mu_x - S_{xx}^{-1}\boldsymbol{S}_{x\pi}(\pi(\underline{x}) - \boldsymbol{\mu}_\pi), S_{xx}^{-1}) \\
&= \mathrm{p}(\pi(\underline{x}))\,\mathrm{p}(\,\underline{x}\,|\,\pi(\underline{x})\,),
\end{aligned}$$

with

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_\pi \\ \mu_x \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{\pi\pi} & \boldsymbol{\Sigma}_{\pi x} \\ \boldsymbol{\Sigma}_{x\pi} & \Sigma_{xx} \end{pmatrix} \text{ and } \boldsymbol{S} = \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \boldsymbol{S}_{\pi\pi} & \boldsymbol{S}_{\pi x} \\ \boldsymbol{S}_{x\pi} & S_{xx} \end{pmatrix}.$$

This results in the following expressions for the conditional variance and the coefficients in Equation 5.13: $\sigma_x^2 = S_{xx}^{-1}$, $\beta_0 = \mu_x$ and $(\alpha_1, \ldots, \alpha_p) = -\sigma_x^2 \boldsymbol{S}_{x\pi}$. We can repeat the same procedure for the remaining multivariate distribution $\mathrm{p}(\pi(\underline{x}))$ iteratively to produce the Bayesian network factorization. Note that the non-zero elements in $\boldsymbol{S}_{x\pi}$ correspond to the parents of node $\underline{x}$. Only the parents for node $\underline{x}$ can be read in this way directly from $\boldsymbol{S} = \boldsymbol{\Sigma}^{-1}$. To go to the next level, we have to look at $\boldsymbol{\Sigma}_{\pi\pi}^{-1}$, which is *not* the same as $\boldsymbol{S}_{\pi\pi}$ in general.

Prior for the parameters

The parameters for this type of local dependency models, are the coefficients β_0 and $(\alpha_1, \ldots, \alpha_p)$ and the conditional variances σ^2. As seen in the previous section, we can equally well use the multivariate Gaussian parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. It is common to work with the precision matrix $\boldsymbol{S} = \boldsymbol{\Sigma}^{-1}$. We will use a Gaussian-Wishart prior for these parameters [23]:

$$\begin{aligned}
\mathrm{p}(\underline{\boldsymbol{S}}) &= \mathrm{C}^{\underline{\mathrm{st}}}\,|\underline{\boldsymbol{S}}|^{(f_0-v-1)/2}\,e^{-\frac{1}{2}tr(\boldsymbol{T}_0\underline{\boldsymbol{S}})} \\
\mathrm{p}(\,\underline{\boldsymbol{\mu}}\,|\,\underline{\boldsymbol{S}}\,) &\sim \mathcal{N}(\boldsymbol{\mu}_0, (\gamma_0\underline{\boldsymbol{S}})^{-1}).
\end{aligned}$$

The precision matrix $\boldsymbol{S}$ has a Wishart distribution with $f_0 > v - 1$ degrees of freedom and a precision matrix $\boldsymbol{T}_0$ which has to be positive definite. The operator $tr(\boldsymbol{X})$ is the trace of a matrix, the sum of its eigenvalues. The mean has a Gaussian distribution with prior mean $\boldsymbol{\mu}_0$ and precision matrix $\gamma_0\boldsymbol{S}$.

Posterior for the parameters

This Gaussian-Wishart prior distribution is conjugate for Gaussian sampling. This means that the a posteriori distribution for $\underline{\boldsymbol{\mu}}$ and $\underline{\boldsymbol{S}}$ after observing a data set $\mathbf{D}$ with n records is again a Gaussian-Wishart distribution with updated parameters

$$\begin{aligned} f_n &= f_0 + n \\ \boldsymbol{T}_n &= \boldsymbol{T}_0 + \hat{\boldsymbol{\Sigma}}_n + \frac{\gamma_0 n}{\gamma_0 + n}(\boldsymbol{\mu}_0 - \hat{\boldsymbol{\mu}}_n)(\boldsymbol{\mu}_0 - \hat{\boldsymbol{\mu}}_n)^T \\ \gamma_n &= \gamma_0 + n \\ \boldsymbol{\mu}_n &= \frac{\gamma_0 \boldsymbol{\mu}_0 + n\hat{\boldsymbol{\mu}}_n}{\gamma_0 + n}. \end{aligned}$$

The symbols $\hat{\boldsymbol{\mu}}_n$ and $\hat{\boldsymbol{\Sigma}}_n$ are the corresponding sample mean and sample covariance matrix of $\mathbf{D}$. The degrees of freedom γ_0 and f_0 can be thought of as equivalent sample sizes.

Probability of local substructure

Using the results from [33], we can write out $L(\,x \,|\, \pi\,)$ as

$$L(\,x \,|\, \pi\,) = \frac{\mathrm{p}(\,\mathbf{D}^{x\pi} \,|\, \mathcal{S}_{\mathrm{bn}}^{\mathrm{C}}\,)}{\mathrm{p}(\,\mathbf{D}^{\pi} \,|\, \mathcal{S}_{\mathrm{bn}}^{\mathrm{C}}\,)}.$$

$\mathbf{D}^{x\pi}$ is the restriction of the data set $\mathbf{D}$ to only the fields x and π, while $\mathcal{S}_{\mathrm{bn}}^{\mathrm{C}}$ denotes the completely connected network structure. If some data set $\mathbf{D}$ comes from a model with a Gaussian-Wishart hyper-distribution, $\mathbf{D}^{x\pi}$ will come from a model with hyper-parameters restricted in the same way. This allows us to write out $L(\,x \,|\, \pi\,)$ as

$$L(\,x \,|\, \pi\,) = \pi^{-n/2} \sqrt{\frac{\gamma_0}{\gamma_0 + n}} \frac{\Gamma(\frac{f_0+n+\#\pi+1}{2})}{\Gamma(\frac{f_0-\#\pi-1}{2})} \left(\frac{|\boldsymbol{T}_0{}^{x\pi}|}{|\boldsymbol{T}_0{}^{\pi}|}\right)^{f_0/2} \left(\frac{|\boldsymbol{T}_n{}^{x\pi}|}{|\boldsymbol{T}_n{}^{\pi}|}\right)^{-(f_0+n)/2}.$$

Unfortunately, the notation π to indicate *parents* is conflicting with the famous mathematical constant describing the half of the circumference of the unit circle. In the above formula, only the factor $\pi^{-n/2}$ has nothing to do with parents.

5.3.3 Nonlinear-Gaussian distributed variables

Although nice results can be obtained using linear-Gaussian variables, we are restricted to the multivariate Gaussian distribution.

Conditional distribution

If this limitation is too severe, we can use more flexible models to describe the conditional mean, instead of a linear one. Here, we will deal with the case where the conditional mean is given by some function $\mathrm{E}[\,\underline{x} \,|\, \pi(\underline{x})\,] = f(\,\pi(\underline{x}) \,|\, \boldsymbol{\nu}\,)$ of the parents with parameters $\boldsymbol{\nu}$

$$\mathrm{p}(\,\underline{x} \,|\, \pi(\underline{x})\,) \sim \mathcal{N}(f(\,\pi(\underline{x}) \,|\, \boldsymbol{\nu}\,), \sigma_x^2).$$

Instead of using the expression of Equation 5.9 where we integrate using the prior parameter distribution, we follow another path: this integration is only possible with a few special distributions, like the table or the linear-Gaussian distribution. We denote with $\mathcal{D}_i$ the first $i{-}1$ data records: $\mathcal{D}_i = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{i-1}\}$. We can decompose the probability of a certain network structure $\mathcal{S}_{\mathrm{bn}}$ given a data set $\mathbf{D}$ by applying the chain rule of probability on the data records:

$$\begin{aligned}
\mathrm{p}(\, \mathcal{S}_{\mathrm{bn}} \,|\, \mathbf{D}\,) &= \frac{\mathrm{p}(\, \mathbf{D} \,|\, \mathcal{S}_{\mathrm{bn}}\,)\, \mathrm{p}(\mathcal{S}_{\mathrm{bn}})}{\mathrm{p}(\mathbf{D})} \\
&\propto \prod_{i=1}^{n} \mathrm{p}(\, \boldsymbol{x}^i \,|\, \mathcal{D}_i, \mathcal{S}_{\mathrm{bn}}\,) \\
&= \prod_{i=1}^{n} \prod_{j=1}^{v} \mathrm{p}(\, x_j^i \,|\, \pi_j^i, \mathcal{D}_i, \mathcal{S}_{\mathrm{bn}}\,) \\
&= \prod_{j=1}^{v} \prod_{i=1}^{n} \mathrm{p}(\, x_j^i \,|\, \pi_j^i, \mathcal{D}_i, \mathcal{S}_{\mathrm{bn}}\,) \\
&= \prod_{j=1}^{v} L(\, x_j \,|\, \pi_j\,).
\end{aligned}$$

The predictions for sample i can be approximated with the maximum a posteriori parametrization for the data subset $\mathcal{D}_i$

$$\mathrm{p}(\, x_j^i \,|\, \pi_j^i, \mathcal{D}_i\,) \approx \mathrm{p}(\, x_j^i \,|\, \pi_j^i, \boldsymbol{\nu}^*\,) \;\text{ with }\; \boldsymbol{\nu}^* = \mathrm{argmax}_{\boldsymbol{\nu}}(\mathrm{p}(\, \boldsymbol{\nu} \,|\, \mathcal{D}_i\,)).$$

Computing $L(\, x \,|\, \pi\,)$ in this way is computationally expensive because it needs n optimizations. This computational cost can be reduced with a factor t by searching a new maximum a posteriori parametrization only every t times, and using this to make the next t predictions [62].

Another method to compute the local substructure probabilities is based on the Monte Carlo method, the posterior distribution, and Equation 5.9:

$$\begin{aligned}
L(\, x \,|\, \pi\,) &= \int \prod_{i=1}^{n} \mathrm{p}(\, x^i \,|\, \pi^i, \boldsymbol{\nu}\,)\, \mathrm{p}(\boldsymbol{\nu})\, d\boldsymbol{\nu} \qquad (5.14) \\
&= \int \prod_{i=1}^{n} \mathrm{p}(\, x^i \,|\, \pi^i, \boldsymbol{\nu}\,)\, \mathrm{p}(\boldsymbol{\nu}) \frac{\mathrm{p}(\, \boldsymbol{\nu} \,|\, \mathbf{D}^{x\pi}\,)}{\mathrm{p}(\, \boldsymbol{\nu} \,|\, \mathbf{D}^{x\pi}\,)} d\boldsymbol{\nu} \\
&\approx \frac{1}{N} \sum_{j=1}^{N} \prod_{i=1}^{n} \mathrm{p}(\, x^i \,|\, \pi^i, \boldsymbol{\nu}_j\,) \frac{\mathrm{p}(\boldsymbol{\nu}_j)}{\mathrm{p}(\, \boldsymbol{\nu}_j \,|\, \mathbf{D}^{x\pi}\,)} \;\text{ with }\; \boldsymbol{\nu}_j \sim \mathrm{p}(\, \boldsymbol{\nu} \,|\, \mathbf{D}^{x\pi}\,).
\end{aligned}$$

Generating parametrizations from the posterior distribution can be done with a Markov chain method, as explained in Section 6.7. We could equally well have based our Monte Carlo summation on the prior distribution $\mathrm{p}(\boldsymbol{\nu})$, and thus have started from Equation 5.14. Unfortunately, $\prod_i \mathrm{p}(\, x^i \,|\, \pi^i, \boldsymbol{\nu}\,)$ will be almost always zero for parameters $\boldsymbol{\nu}$ drawn from the prior distribution. The prior distribution is simply too broad and does not contain any information about the problem. Expanding the integral with a Monte Carlo summation based on the posterior distribution will help, but complicates the sampling procedures often.

5.3.4 Missing values and hidden variables

Until now, we always assumed completely observed data. In practice, this is not very often the case as some values may be missing from the data set or even worse, each value for a certain variable may be missing. The first type is called *missing* values, while the latter involves a data set with *hidden* variables [71].

Missing values can be dealt with using the Gibbs sampling procedure [76]. If we denote with $\mathbf{D}^{\mathrm{O}}$ the observed part of the data set, and with $\mathbf{D}^{\mathrm{M}}$ the missing part, we can repeat the following steps in an iterative manner:

1. $\mathrm{p}(\,\mathbf{D}^{\mathrm{M}}{}_{i+1} \,|\, \mathcal{S}_{\mathrm{bn}i}, \boldsymbol{\nu}_i, \mathbf{D}^{\mathrm{O}}\,) = \mathrm{p}(\,\mathbf{D}^{\mathrm{M}} \,|\, \mathcal{S}_{\mathrm{bn}i}, \boldsymbol{\nu}_i\,)$
2. $\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}i+1} \,|\, \mathbf{D}^{\mathrm{O}}, \mathbf{D}^{\mathrm{M}}{}_{i+1}\,)$
3. $\mathrm{p}(\,\boldsymbol{\nu}_{i+1} \,|\, \mathbf{D}^{\mathrm{O}}, \mathbf{D}^{\mathrm{M}}{}_{i+1}\,)$.

The initial network structure and parameters can be learned using only the complete records. Since there are no hidden variables, there are always some samples that are completely observed, we can obtain a sensible start point for our procedure using these completely observed samples. This Gibbs sampling method will eventually generate samples according to the joint distribution

$$\mathrm{p}(\,\mathbf{D}^{\mathrm{M}}, \boldsymbol{\nu}, \mathcal{S}_{\mathrm{bn}} \,|\, \mathbf{D}^{\mathrm{O}}\,).$$

More interesting is dealing with *hidden variables* because such models have a wide application. A model with one hidden node that is the parent of all the other nodes, represents a clustered distribution: depending on the value of this hidden node, and possibly other nodes, a different distribution for the child nodes can be used. By learning the structure of such a model where the hidden node can, but not necessarily is, connected to the other nodes, biclustering will be performed.

To learn the parameters for a given structure and a partially observed data set, a good choice is to use *variational learning* [51, 27]. These techniques basically try to perform inference for the missing values or hidden variables $\underline{\boldsymbol{x}}^{\mathrm{M}}$, given the network structure $\mathcal{S}_{\mathrm{bn}}$ and parameters $\boldsymbol{\nu}$. This inference is nothing more than trying to find the distribution of $\underline{\boldsymbol{x}}^{\mathrm{M}}$ given the observed data $\underline{\boldsymbol{x}}^{\mathrm{O}}$, the network structure, and it corresponding parameters:

$$\mathrm{p}(\,\underline{\boldsymbol{x}}^{\mathrm{M}} \,|\, \underline{\boldsymbol{x}}^{\mathrm{O}}, \mathcal{S}_{\mathrm{bn}}, \boldsymbol{\nu}\,).$$

This distribution can be found using the variational bound [65]:

$$\begin{aligned} F(q(\cdot), \mathcal{S}_{\mathrm{bn}}, \boldsymbol{\nu}) &= \mathrm{E}_q[\log(\mathrm{p}(\,\underline{\boldsymbol{x}}^{\mathrm{M}}, \underline{\boldsymbol{x}}^{\mathrm{O}} \,|\, \mathcal{S}_{\mathrm{bn}}, \boldsymbol{\nu}\,))] - \mathrm{E}_q[\log(q(\underline{\boldsymbol{x}}^{\mathrm{M}}))] \\ &\leq \log(\mathrm{p}(\underline{\boldsymbol{x}}^{\mathrm{O}})). \end{aligned} \tag{5.15}$$

The bound $F(q(\cdot), \mathcal{S}_{\text{bn}}, \boldsymbol{\nu})$ depends on a distribution $q(\cdot)$ over the missing variables $\underline{\boldsymbol{x}}^{\text{M}}$, the network structure and corresponding parameters, and appears to be bounded by the right hand side of the above expression. The average $\text{E}_q[\log(\text{p}(\,\underline{\boldsymbol{x}}^{\text{M}}, \underline{\boldsymbol{x}}^{\text{O}} \,|\, \mathcal{S}_{\text{bn}}, \boldsymbol{\nu}\,))]$ is over the missing values $\underline{\boldsymbol{x}}^{\text{M}}$, using the distribution $q(\cdot)$. $F(\cdot)$ consists of two parts: the first term measures how well the values drawn from $q(\cdot)$ fit within the joint distribution $\text{p}(\,\underline{\boldsymbol{x}} \,|\, \mathcal{S}_{\text{bn}}, \boldsymbol{\nu}\,)$. The second term is the entropy of the distribution $q(\cdot)$ and tries to keep is as broad and uninformative as possible.

If the multivariate distribution $q(\cdot)$ is *unconstrained*, this bound is maximized for $q(\underline{\boldsymbol{x}}^{\text{M}}) = \text{p}(\,\underline{\boldsymbol{x}}^{\text{M}} \,|\, \underline{\boldsymbol{x}}^{\text{O}}, \mathcal{S}_{\text{bn}}, \boldsymbol{\nu}\,)$:

$$\text{p}(\,\underline{\boldsymbol{x}}^{\text{M}} \,|\, \underline{\boldsymbol{x}}^{\text{O}}, \mathcal{S}_{\text{bn}}, \boldsymbol{\nu}\,) = \text{argmax}_{q(\cdot)}(F(q(\cdot), \mathcal{S}_{\text{bn}}, \boldsymbol{\nu})).$$

For this special $q(\cdot)$, Equation 5.15 becomes an equality. Maximizing this bound is the same as performing inference.

The variational technique consists of restricting the $q(\cdot)$ to a certain family of distributions. Optimizing the bound under this restriction will perform only *approximate* inference, but removes the impossible task of optimizing over *all* possible distributions. Although Frey et al. [27] use independent Gaussians to approximate $q(\cdot)$, we will choose independent univariate Laplace distributions as the family of distributions:

$$\begin{aligned} q(\underline{\boldsymbol{x}}^{\text{M}}) &= \prod_i q(\underline{\boldsymbol{x}}_i^{\text{M}}) \\ &= \prod_i \frac{1}{\sqrt{2}\sigma_i} e^{-\frac{1}{\sqrt{2}\sigma_i}|\underline{\boldsymbol{x}}_i^{\text{M}} - \mu_i|}. \end{aligned}$$

If we use these distributions in combination with the nonlinear Laplace transfer function

$$\phi(x) = 2\int_0^x \frac{1}{\sqrt{2}} e^{-\frac{1}{\sqrt{2}}|y|} dy = \text{sign}(x)(1 - e^{-\sqrt{2}|x|}),$$

we can use the techniques described in [27] more efficiently because we can compute the needed output variance exactly.

The above described technique can perform (approximate) inference using a *fixed* network structure and a data set with hidden variables. The results of this inference (the distribution $q(\cdot)$) can then be used to generate values for the missing part of the data set. Once the data set is completed, a complete-data structure learning method can be used to update the structure, and new inference can be performed and the whole procedure can be repeated. A similar technique called structural EM is described by Friedman [28].

5.4 Different types of prior information available

Now that we have introduced the concept of Bayesian networks together with some algorithms to learn and use them, we can place them in the transformation framework we discussed in Chapter 4. As indicated before, these models consist of two components: the network structure and the local dependency models. The structure has a clear interpretation in terms of conditional independency statements. Depending on the choice of the local dependency models, these conditional probability models can have a clear interpretation too. If this is the case, the model is called a white box model because we can reason about and interpret its components.

To discuss a few possible sources of prior information, we will assume discrete domain variables and table distributed dependency models. We will use the medical example introduced in Chapter 2 to illustrate and motivate the methods.

5.4.1 Expert information

At first, we look at the information that is contained in a *domain expert*. This domain expert is someone who is familiar with the domain and has experience with the variables and their dependencies.

For our task of pre-operative classification of ovarian tumours, our expert is Prof. Dr. Dirk Timmerman, a medical doctor specialized in this domain. He has examined many patients and has a clear understanding of the relationships between the domain variables that are involved, and how these relationships look like.

5.4.2 Textual information

Another source of information that is frequently available are written documents. These can be documents describing *one* specific domain variable, or general documents describing some part of the domain.

For our problem, documents of the first kind consist of the IOTA protocols, a description for each variable from a thesis [81], and definitions from the Merck Manual. In addition, we added specific documents from the On-line Medical Dictionary, the CancerNet Dictionary and the MEDLINE collection of abstracts of the US National Library of Medicine to these descriptions. We selected three gradually increasing subsets of the MEDLINE abstracts as documents describing part of the domain. These subsets contain respectively 5 367, 71 845 and 378 082 documents respectively and date from the period January 1982 until November 2000.

5.5 Prior information representation

Now that we know what kind of information we have access to, we have to convert it to a format that is suitable to perform calculations. We will introduce these formats to make clear what kind of information will be necessary.

5.5.1 Prior for the structure

We start with a prior distribution over the space of network structures. There are basically two approaches to define such a prior distribution. The first approach makes use of a certain fixed network structure $\mathcal{S}^*_{\mathrm{bn}}$ and defines the probability of another structure $\mathcal{S}_{\mathrm{bn}}$ based on the distance between the two structures. This distance in network space is based on counting the number of arc removals, additions, and reversals that are necessary to modify $\mathcal{S}_{\mathrm{bn}}$ to $\mathcal{S}^*_{\mathrm{bn}}$. The other possibility basically assumes that the probability of a network structure can be decomposed in a series of independent edge probabilities:

$$\begin{aligned}
\mathrm{p}(\mathcal{S}_{\mathrm{bn}}) &= \prod_{\underline{x}} \mathrm{p}(\pi(\underline{x}) \to \underline{x}) \\
\mathrm{p}(\pi(\underline{x}) \to \underline{x}) &= \prod_{\underline{y} \in \pi(\underline{x})} \mathrm{p}(\underline{y} \to \underline{x}) \prod_{\underline{y} \notin \pi(\underline{x})} (1 - \mathrm{p}(\underline{y} \to \underline{x})).
\end{aligned}$$

The notation $\mathrm{p}(\pi(\underline{x}) \to \underline{x})$ indicates the probability of the local substructure where variable $\underline{x}$ has parents $\pi(\underline{x})$.

The parameters for this last type of distribution are the individual arc probabilities, which can be represented in a matrix. In Section 5.5.3, we will be able to more or less derive a matrix $\boldsymbol{V}$ from the expert or the literature information where the elements represent the *connectedness* between variables. Rather than using these values immediately as arc probabilities, we will introduce an extra parameter ν which controls the density of the networks that will be generated from the distribution; we will transform all the matrix elements with an exponent ζ such that the average of the mean number of parents per local substructure is ν:

$$\begin{aligned}
\mathrm{p}(\underline{y} \to \underline{x}) &= \boldsymbol{V}^{\zeta}_{\underline{x}\underline{y}} \\
\nu &= \frac{1}{v} \sum_{\underline{x}} \mathrm{E}[\#\pi(\underline{x})] \\
&= \frac{1}{v} \sum_{\underline{x}} \sum_{\underline{y}} \mathrm{p}(\underline{y} \to \underline{x}) \\
&= \frac{1}{v} \sum_{\underline{x}} \sum_{\underline{y}} \boldsymbol{V}^{\zeta}_{\underline{x}\underline{y}}.
\end{aligned}$$

Finding the exponent ζ that gives rise to the correct mean number of parents can be done with any single variable optimization algorithm. With this mean number of parents, we can control the complexity of the networks that will be learned. Note that the distribution we define, is over the set of directed graphs. For Bayesian network applications, we will restrict this distribution to the set of directed *acyclic* graphs.

If necessary, both distributions can be combined to one. Both of these priors decompose into a product of factors where each factor describes the probability of one variable and its parents

$$\mathrm{p}(\mathcal{S}_{\mathrm{bn}}) = \prod_{\underline{x}} \mathrm{p}(\pi(\underline{x}) \to \underline{x}),$$

which makes it possible to use the structure learning procedures described in Section 5.2.1 by introducing a new local substructure score $L^*(\,\underline{x}\,|\,\pi(\underline{x})\,)$:

$$\begin{aligned}
\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}}\,|\,\mathbf{D}\,) &\propto \mathrm{p}(\mathcal{S}_{\mathrm{bn}})\,\mathcal{L}(\,\mathbf{D}\,|\,\mathcal{S}_{\mathrm{bn}}\,) \\
&= \prod_{\underline{x}} \mathrm{p}(\pi(\underline{x}) \to \underline{x}) \prod_{\underline{x}} L(\,\underline{x}\,|\,\pi(\underline{x})\,) \\
&= \prod_{\underline{x}} \mathrm{p}(\pi(\underline{x}) \to \underline{x})\, L(\,\underline{x}\,|\,\pi(\underline{x})\,) \\
&= \prod_{\underline{x}} L^*(\,\underline{x}\,|\,\pi(\underline{x})\,).
\end{aligned}$$

5.5.2 Prior for the parameters

Once the prior over the model structures is specified, we would like to have a parameter prior for each of these structures. In the discrete case, this corresponds to specifying the Dirichlet hyper-parameters m_{jkl} for each network structure. Heckerman [34] suggests to specify the Bayesian network parameters $\boldsymbol{\theta}^*$ only for the fully connected network $\mathcal{S}_{\mathrm{bn}}^{\mathrm{C}}$, in combination with an equivalent sample size M:

$$m_{jkl} = M \cdot \mathrm{p}(\,\underline{x}_j = l\,|\,\pi(\underline{x}_j) = \pi_k, \boldsymbol{\theta}^*, \mathcal{S}_{\mathrm{bn}}^{\mathrm{C}}\,).$$

The Bayesian network model $(\boldsymbol{\theta}^*, \mathcal{S}_{\mathrm{bn}}^{\mathrm{C}})$ is a point estimate of how the values of the different variables depend on each other. The equivalent sample size expresses the value of this point estimate in terms of number of samples. The hyper-parameters m_{jkl} that result from this procedure are the same as the posterior Dirichlet parameters n_{jkl} that would result from a data set with M samples and empirical probabilities equal to $\mathrm{p}(\,\underline{x}_j = l\,|\,\pi(\underline{x}_j) = \pi_k, \boldsymbol{\theta}^*, \mathcal{S}_{\mathrm{bn}}^{\mathrm{C}}\,)$.

Because specifying the parameters for a fully connected network is a daunting task, we will ask our expert to construct a network $\mathcal{S}_{\mathrm{bn}}^*$ that faithfully represents the structure of the domain but for which it is feasible to generate a parametrization also. Doing so, any necessary Dirichlet hyper-parametrization can be computed with an inference method for the Bayesian network model $(\boldsymbol{\theta}^*, \mathcal{S}_{\mathrm{bn}}^*)$:

$$m_{jkl} = M \cdot \mathrm{p}(\,\underline{x}_j = l\,|\,\pi(\underline{x}_j) = \pi_k, \boldsymbol{\theta}^*, \mathcal{S}_{\mathrm{bn}}^*\,).$$

5.5.3 Harvesting the information sources

We discuss briefly how the necessary information can be gathered from one of our information sources.

Prior network structure and parametrization from expert

We can ask our expert to construct a network structure $\mathcal{S}^*_{\mathrm{bn}}$ with corresponding parameters $\boldsymbol{\theta}^*$ and an equivalent sample size M. To aid this process, it helps to initially group the variables in semantic classes and order the variables according to their importance. Initially, a small network can be constructed within each group, between groups or between the more important variables. Causal reasoning is similarly a helpful tool to specify a network structure.

Once the structure is completed, its parameters should be specified. These parameters have a direct interpretation in terms of probabilities and can be translated to written questions like:

> Consider a patient with $\pi(\underline{x})_1 = y_1$ and $\pi(\underline{x})_2 = y_2$. How likely is it that $\underline{x} = x$?

The notation $\pi(\underline{x})_i$ denotes the i^{th} parent of variable $\underline{x}$.

Although it is possible to directly ask for numbers, we found it helpful to present our expert with a graphical scale from 0 to 1 where the probability can be indicated upon. This scale includes indications like *almost impossible*, *improbable*, *uncertain*, *fifty-fifty*, *expected*, *probable*, and *almost certain.*

Similarity matrix from expert

To construct the similarity matrix from expert information, we asked to select for each domain variable those other variables that are in relation with each other. We divided these selected variables in three groups: *very important, important*, and *slightly connected.* Then, where it was possible, we tried to refine this division and make the symmetric links consistent.

Similarity matrix from textual information

Instead of asking our expert to generate the similarity matrix, we tried to generate this also from the available textual information. We assume we have a textual description $\mathcal{D}(\underline{x})$ for each domain variable $\underline{x}$ and a large document corpus $\mathcal{C}$ with documents about the domain at hand.

To work with a document $\mathcal{D}$, we first convert it to a vector representation $\mathcal{T}(\mathcal{D})$ where each component represents the weight of a corresponding word, neglecting the grammatical structure of the text. This weight is computed using the popular *term-frequency inverse-document-frequency* (tf-idf) weighting scheme [7]

$$\omega(\circledast \,|\, \mathcal{D}) = -\frac{\#\circledast \in \mathcal{D}}{\#\mathcal{D}} \log\left(\frac{\#\mathcal{C}}{\#\mathcal{C}|_{\circledast}}\right).$$

In this formula, $\omega(\circledast \,|\, \mathcal{D})$ represents the tf-idf weight of the word $\circledast$ in the document $\mathcal{D}$, $\#\mathcal{D}$ is the total number of words in document $\mathcal{D}$, $\#\mathcal{C}$ is the total number of documents and $\#\mathcal{C}|_{\circledast}$ is the number of documents in $\mathcal{C}$ that contain the word $\circledast$. The first fraction is the frequency of the word in the document. The higher this frequency is, the more representative it is for our document.

Unfortunately, some very common words like “the”, “for”, or “and” have a high frequency in *each* document, which makes them useless to discriminate between documents. The second fraction with the logarithm together with the minus, is the inverse document frequency, which weights down these frequent words.

Using the vector representations $\mathcal{T}(\mathcal{D}_1)$ and $\mathcal{T}(\mathcal{D}_2)$ of two documents $\mathcal{D}_1$ and $\mathcal{D}_2$, we can define a similarity measure between them. The most common method is based on the cosine of the angle between the vector representations of these two documents

$$\mathrm{sim}(\mathcal{D}_1, \mathcal{D}_2) = \frac{< \mathcal{T}(\mathcal{D}_1), \mathcal{T}(\mathcal{D}_2) >}{\|\mathcal{T}(\mathcal{D}_1)\|\|\mathcal{T}(\mathcal{D}_2)\|}.$$

Using this similarity measure, we can either directly define the similarity between domain variables as the similarity between their corresponding annotations

$$\boldsymbol{V}_{\underline{x}\underline{y}} = \boldsymbol{V}_{\underline{y}\underline{x}} = \mathrm{sim}(\mathcal{D}(\underline{x}), \mathcal{D}(\underline{y}))).$$

Although the previous method works nicely, we only use our corpus of documents to filter out the frequent words. We can use this corpus more extensively using the following technique: suppose that the annotation $\mathcal{D}(\underline{x})$ for each variable consists only of a few keywords. We define the connectedness between two variables $\underline{x}$ and $\underline{y}$ as the probability that a document in our corpus $\mathcal{C}$ discusses both variables given that the document discusses one of them

$$\boldsymbol{V}_{\underline{x}\underline{y}} = \boldsymbol{V}_{\underline{y}\underline{x}} = \mathrm{P}(\,\underline{x} \in \mathcal{D} \text{ and } \underline{y} \in \mathcal{D} \,|\, \underline{x} \in \mathcal{D} \text{ or } \underline{y} \in \mathcal{D}, \mathcal{D} \in \mathcal{C}\,).$$

The notation $\underline{x} \in \mathcal{D}$ means that the keywords of the annotation $\mathcal{D}(\underline{x})$ are found in $\mathcal{D}$. This method can be extended using the tf-idf vector representation instead of the keywords describing the variables.

Note that in practice there are some important pre-processing steps that are necessary. As such, we have to convert each word to its canonical form using a stemmer. Domain specific words, phrases, and synonyms also need special treatment.

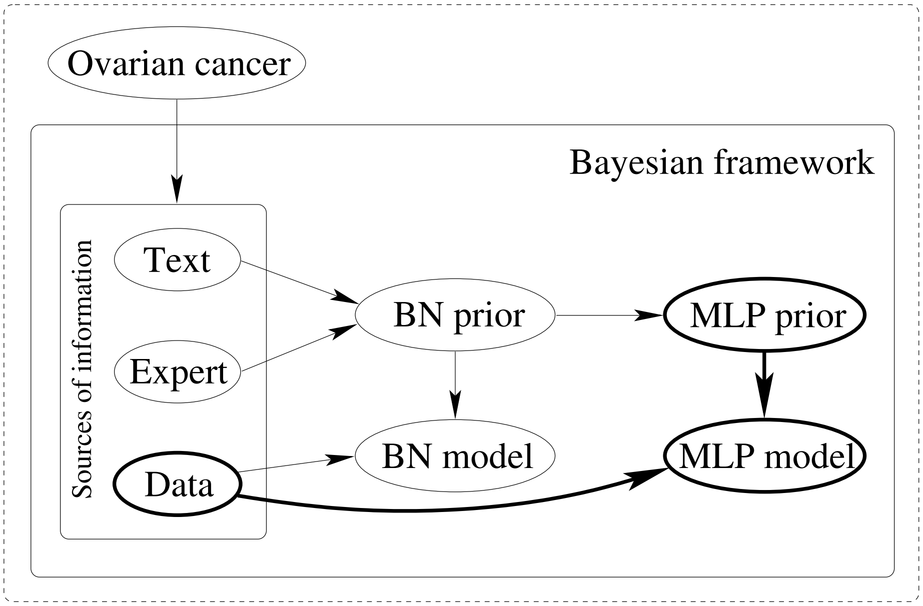
5.6 To conclude

The concept of representing joint distributions using Bayesian networks dates from the eighties [68, 58, 44]. These types of networks became popular in artificial intelligence and machine learning communities, resulting in a nice collection of algorithms and procedures to perform various tasks with them. The most important of them were introduced in this chapter. The practical application of Bayesian networks lagged a bit behind initially [29], but now they have become a popular tool to model various types of problems in genetics, medicine, and many more application areas.

We contributed to the applicability of Bayesian networks by defining a probability distribution over the space of Bayesian network structures, both based on expert information or text documents. To facilitate this process, we developed or extended a few tools, like finding a good ancestral node ordering, the visual layout generation of Bayesian networks based on self-organising maps, or the extension of the variational inference technique using Laplace distributions.

Chapter 6

Multilayer perceptrons



In the previous chapter, we introduced the Bayesian network model class, which will serve as the donor model class for the transformation technique presented in Chapter 4. In this chapter, we will introduce the acceptor model class. *Once this last model class is introduced, we are ready to perform actual experiments.*

The model class we will introduce now, is the class of multilayer perceptrons. *These models are called black box models as there is no or little insight in their parameter or structure setting in the general case. Despite this drawback, they have good learning capabilities from data, which makes them popular models.*

In contrast with a Bayesian network that models a joint distribution, a multilayer perceptron will only model a conditional distribution by providing its conditional mean in the regression case, or by providing a model for class membership probability in the classification case. Multilayer perceptrons can indeed be seen as an extension to linear logistic regression models, where the conditional class probability is now provided by the input-output mapping described by the multilayer perceptron. Since our main application is focused on classification, we will most often use multilayer perceptrons in the logistic regression context.

6.1 Description

A multilayer perceptron is a parametrized function from $\mathbb{R}^d$ to $\mathbb{R}$. It can be represented graphically by a layered, feed-forward network of neurons, as shown in Figure 6.1.

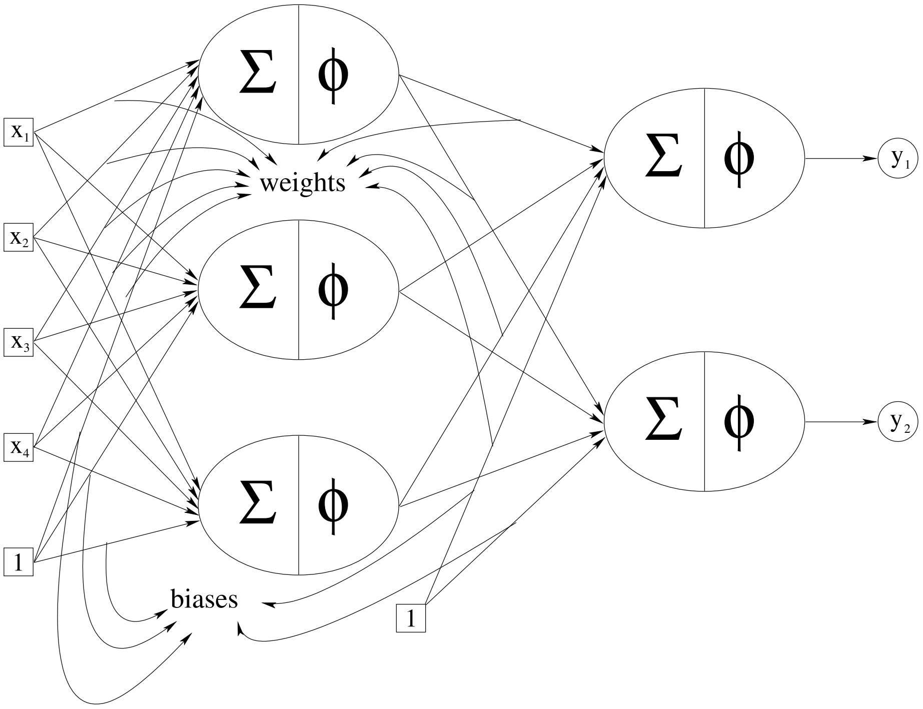


Figure 6.1: A multilayer perceptron structure with one hidden layer. The weights are the arcs from the inputs to the first layer and from the first layer to the second layer. The biases are the weights from the inputs equal to 1.

Each neuron computes the weighted sum of the outputs of the previous layer, adds a bias term to it, performs a nonlinear squashing function $\phi(.)$ on this sum, and passes the value again to the next layer:

$$\mathrm{mlp}(\,\boldsymbol{x} \,|\, \boldsymbol{\omega}\,) = \phi(\ldots\phi(\sum_j \phi(\sum_i x_i \boldsymbol{\omega}^1_{ji} + \boldsymbol{\omega}^1_{jb})\boldsymbol{\omega}^2_{kj} + \boldsymbol{\omega}^2_{kb})\ldots).$$

The inputs of this multilayer perceptron are indicated by $\boldsymbol{x} = (x_1, \ldots, x_d)$. The parameters of this mapping are denoted with the symbol $\boldsymbol{\omega}$, and are the weights of the weighted sums that are at the heart of the multilayer perceptron. These parameters are often split into two groups, the weights that make the connection with the inputs or nodes from the previous layer (e.g., $\boldsymbol{\omega}^1_{ji}$ or $\boldsymbol{\omega}^2_{kj}$), and the additive weights like $\boldsymbol{\omega}^1_{jb}$ or $\boldsymbol{\omega}^2_{kb}$, which are indicated with the name *bias*. The bias is seen as the weight of a constant input equal to one, and is always included if we speak about the weights, unless mentioned otherwise. The bias term allows a translation before the transfer function is applied. The weights are the function parameters and determine the shape of the mapping.

The nonlinear squashing function, or transfer function, can be anything. Popular choices are the hyperbolic tangent, the sigmoidal function, or the sign function. For symmetry reasons, as well as its connection to logistic regression (see Section 6.2.3), we will always use the hyperbolic tangent transfer function

$$\phi(x) \equiv \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{2}{1 + e^{-2x}} - 1.$$

Its domain is $\mathbb{R}$, its range is $]-1, 1[$ and its graph can be seen on Figure 6.2. The derivative of $\phi(.)$ can be expressed in terms of $\phi(.)$ itself:

$$\phi'(x) = (1 + \phi(x))\,(1 - \phi(x)).$$

This differential equation could equally well be used as the very definition of the hyperbolic tangent function.

6.2 Conditional probability models and cost functions

Using multilayer perceptrons as a class of functions that has the universal approximation property (see Section 6.3), we will define conditional probability models. The goal is to describe the distribution of the response variable $\underline{y}$ (also called the dependent variable) conditioned on a set of covariates (also called the independent variables). Note that the name *independent variables* is confusing since it does not mean they are statistically independent. Two common methods are based on either using a multilayer perceptron as a model for the conditional mean or using the output of the network directly as the probability that the dependent variable belongs to some class in a classification setup. Such a conditional distribution is directly related to a cost or error function, a performance measure that is often used to optimize the weights of a network.

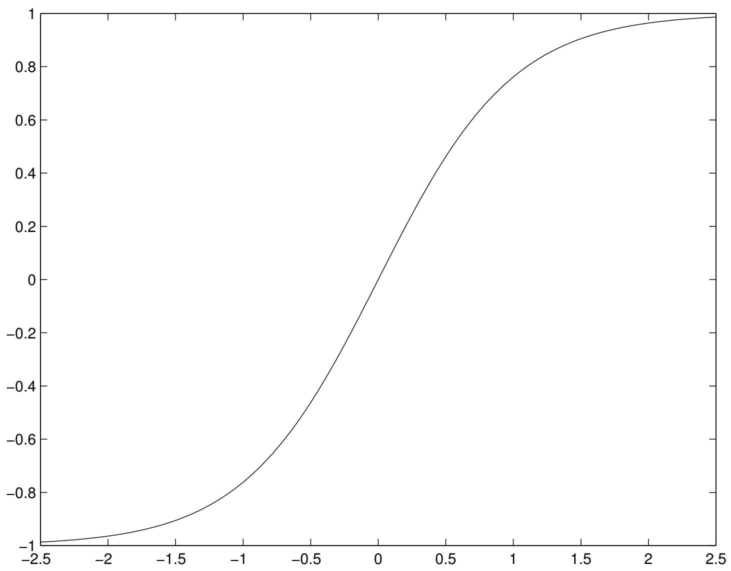


Figure 6.2: The hyperbolic tangent transfer function $\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

6.2.1 Regression

In the regression framework, the output of a neural network is used as a model for the mean of $\underline{y}$ conditioned on $\underline{\boldsymbol{x}}$:

$$\mathrm{E}[\,\underline{y}\,|\,\underline{\boldsymbol{x}}\,] = \mathrm{mlp}(\,\underline{\boldsymbol{x}}\,|\,\boldsymbol{\omega}\,).$$

Depending on the distribution that has to be modelled, the distribution around this mean has to be chosen. A common choice is a Gaussian distribution with a fixed variance σ^2 independent of $\underline{\boldsymbol{x}}$:

$$\mathrm{p}(\,\underline{y}\,|\,\underline{\boldsymbol{x}}, \boldsymbol{\omega}\,) = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{1}{2\sigma^2}(\underline{y} - \mathrm{mlp}(\,\underline{\boldsymbol{x}}\,|\,\boldsymbol{\omega}\,))^2}.$$

If we observe some data set $\mathbf{D}$, we can define a cost or error function based on the likelihood of the network parameters $\boldsymbol{\omega}$ given the data:

$$\begin{aligned}
\mathrm{Error}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) &= -\log(\mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)) \\
&= -\log(\mathrm{p}(\,\mathbf{D}\,|\,\boldsymbol{\omega}\,)) \\
&= -\log(\prod_{i=1}^{n} \mathrm{p}(\,\underline{\boldsymbol{x}}_i, \underline{y}_i\,|\,\boldsymbol{\omega}\,)) \\
&= -\sum_{i=1}^{n} \log(\mathrm{p}(\,\underline{y}_i\,|\,\boldsymbol{\omega}, \underline{\boldsymbol{x}}_i\,)\,\mathrm{p}(\,\underline{\boldsymbol{x}}_i\,|\,\boldsymbol{\omega}\,)) \\
&= -\sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) + \sum_{i=1}^{n} \frac{1}{2\sigma^2}\,(\underline{y}_i - \mathrm{mlp}(\,\underline{\boldsymbol{x}}_i\,|\,\boldsymbol{\omega}\,))^2 \\
&\quad -\sum_{i=1}^{n} \log(\mathrm{p}(\,\underline{\boldsymbol{x}}_i\,|\,\boldsymbol{\omega}\,)) \\
&= \sum_{i=1}^{n} \frac{1}{2\sigma^2}(\underline{y}_i - \mathrm{mlp}(\,\underline{\boldsymbol{x}}_i\,|\,\boldsymbol{\omega}\,))^2 + \mathrm{C}^{\underline{\mathrm{st}}}.
\end{aligned}$$

The term $-\sum_{i=1}^{n}\log(\frac{1}{\sqrt{2\pi}\sigma})$ is independent of the network parameters $\boldsymbol{\omega}$. Although less obvious, the term $-\sum_{i=1}^{n}\log(\mathrm{p}(\,\underline{\boldsymbol{x}}_i\,|\,\boldsymbol{\omega}\,))$ is also independent of $\boldsymbol{\omega}$: the multilayer perceptron models *only* the conditional distribution $\mathrm{p}(\,\underline{y}\,|\,\underline{\boldsymbol{x}}\,)$ and says nothing about about the distribution of the inputs $\underline{\boldsymbol{x}}$, which allows us to write $\mathrm{p}(\,\underline{\boldsymbol{x}}_i\,|\,\boldsymbol{\omega}\,) = \mathrm{p}(\underline{\boldsymbol{x}}_i)$. Because these terms are independent, they can be dropped if the purpose is to optimize the error function.

This error function is called the sum-of-squares error for obvious reasons. Minimizing this function results in the least squares solution $\boldsymbol{\omega}^*$, which has the appealing property that the corresponding network function will equal the conditional mean

$$\mathrm{mlp}(\,\underline{\boldsymbol{x}}\,|\,\boldsymbol{\omega}^*\,) = \mathrm{E}[\,\underline{y}\,|\,\underline{\boldsymbol{x}}\,],$$

regardless of the distribution of $\underline{y}$. This statement holds true if the corresponding network structure is flexible enough, sufficient data is present, and the minimization routine is able to find the global minimum $\boldsymbol{\omega}^*$ [11].

If the network structure consists of only one neuron using a linear transfer function, it implements the well-known linear regression model, as will be explained in Section 6.2.3.

6.2.2 Classification

When the response variable is dichotomous, a more appropriate model is required to describe the conditional distribution. Suppose that $\underline{y}$ can be either -1 or 1. We could model the distribution of $\underline{y}$ by interpreting the network output as the probability that $\underline{y} = 1$:

$$\begin{aligned}
\mathrm{P}(\,\underline{y} = 1\,|\,\underline{\boldsymbol{x}}, \boldsymbol{\omega}\,) &= \mathrm{mlp}(\,\underline{\boldsymbol{x}}\,|\,\boldsymbol{\omega}\,) \\
\mathrm{P}(\,\underline{y} = y\,|\,\underline{\boldsymbol{x}}, \boldsymbol{\omega}\,) &= \mathrm{mlp}(\,\underline{\boldsymbol{x}}\,|\,\boldsymbol{\omega}\,)^{\frac{y+1}{2}}(1 - \mathrm{mlp}(\,\underline{\boldsymbol{x}}\,|\,\boldsymbol{\omega}\,))^{\frac{1-y}{2}}.
\end{aligned}$$

This requires the network output to be in the interval [0,1]. This is often achieved by using the sigmoidal transfer function of Equation 6.3, which has the desired domain and other appealing properties, as explained in the following section. For symmetry reasons we will explain in Section 6.3.3, we will use the hyperbolic tangent function instead. Although its domain is [-1,1], this is easily transformed to [0,1], which allows us to interpret the output as a probability anyway:

$$\begin{aligned}
\mathrm{P}(\,\underline{y} = 1 \,|\, \underline{\boldsymbol{x}}, \boldsymbol{\omega}\,) &= \frac{1 + \mathrm{mlp}(\,\underline{\boldsymbol{x}} \,|\, \boldsymbol{\omega}\,)}{2} \\
\mathrm{P}(\,\underline{y} = y \,|\, \underline{\boldsymbol{x}}, \boldsymbol{\omega}\,) &= \left(\frac{1 + \mathrm{mlp}(\,\underline{\boldsymbol{x}} \,|\, \boldsymbol{\omega}\,)}{2}\right)^{\frac{y+1}{2}} \left(\frac{1 - \mathrm{mlp}(\,\underline{\boldsymbol{x}} \,|\, \boldsymbol{\omega}\,)}{2}\right)^{\frac{1-y}{2}} \\
&= \frac{1}{2}(1 + \mathrm{mlp}(\,\underline{\boldsymbol{x}} \,|\, \boldsymbol{\omega}\,))^{\frac{y+1}{2}}(1 - \mathrm{mlp}(\,\underline{\boldsymbol{x}} \,|\, \boldsymbol{\omega}\,))^{\frac{1-y}{2}}. \qquad (6.1)
\end{aligned}$$

The corresponding error function is defined in exactly the same way as for the regression case:

$$\begin{aligned}
\mathrm{Error}(\,\boldsymbol{\omega} \,|\, \mathbf{D}\,) &= -\log(\mathcal{L}(\,\boldsymbol{\omega} \,|\, \mathbf{D}\,)) \\
&= -\log\left(\prod_{i=1}^{n} \mathrm{p}(\,\underline{y}_i \,|\, \underline{\boldsymbol{x}}_i, \boldsymbol{\omega}\,)\mathrm{p}(\,\underline{\boldsymbol{x}}_i \,|\, \boldsymbol{\omega}\,)\right) \\
&= -\sum_{i=1}^{n} \log(1 + \mathrm{sign}(\underline{y}_i)\,\mathrm{mlp}(\,\underline{\boldsymbol{x}}_i \,|\, \boldsymbol{\omega}\,)) + \mathrm{C}^{\underline{\mathrm{st}}}.
\end{aligned}$$

We again define the error function based on the negative logarithm of the likelihood function. The likelihood function can be rewritten as a product over the different data records, and for each data record $\{\underline{y}_i, \underline{\boldsymbol{x}}_i\}$, we can apply Equation 6.1.

6.2.3 Logistic regression

The sigmoidal function originates from the logistic regression model [43]. As opposed to linear regression where a linear model of the covariates $(\underline{x}_1, \ldots, \underline{x}_d)$ tries to predict the conditional mean of the response variable $\underline{y}$, a logistic regression model tries to predict the logarithm of the odds of the response variable with a linear combination of the covariates. The response variable $\underline{y}$ is assumed to be binary and the odds of this variable is defined as $\mathrm{P}(\underline{y} = 1)/(1 - \mathrm{P}(\underline{y} = 1))$. This results in the model

$$\begin{aligned}
\log\left(\frac{\mathrm{P}(\underline{y} = 1)}{1 - \mathrm{P}(\underline{y} = 1)}\right) &= \beta_0 + \beta_1 \underline{x}_1 + \cdots + \beta_v \underline{x}_v \qquad (6.2) \\
&= \mu.
\end{aligned}$$

To find $\mathrm{P}(\underline{y}=1)$, we have to apply the sigmoidal function to Equation 6.2:

$$\begin{aligned}\mathrm{P}(\underline{y}=1) &= \frac{e^{\mu}}{1+e^{\mu}} \\ &= \frac{1}{1+e^{-\mu}} \\ &= \mathrm{sigmoid}(\mu).\end{aligned} \tag{6.3}$$

This function can be transformed to the hyperbolic tangent function using the linear transformation

$$\phi(x) = 2\,\mathrm{sigmoid}(2x) - 1.$$

This property makes the logistic regression model class a subclass of the multilayer perceptron class with hyperbolic tangent transfer functions, as long as we encode the output variable with {-1,1} instead of {0,1}.

Most of the theory of linear regression applies also to logistic regression. We will in particular make use of input selection techniques where one wants to know which covariates should be included and which should be removed from the model [72].

6.3 Approximation and overtraining

Multilayer perceptrons, even with only one hidden layer, are powerful models for approximating continuous functions if we are free to choose the number of hidden neurons of the network. Hornik [42] proved that for each continuous function $f : [0,1]^d \to \mathbb{R}$ and for each error level $\varepsilon > 0$, there exists a multilayer perceptron with parameters $\boldsymbol{\omega}$ with only one hidden hyperbolic tangent layer in its structure $\mathcal{S}_{\mathrm{mlp}}$, such that:

$$\sup_{\boldsymbol{x}\in[0,1]^d} |f(\boldsymbol{x}) - \mathrm{mlp}(\,\boldsymbol{x} \,|\, \mathcal{S}_{\mathrm{mlp}}, \boldsymbol{\omega}\,)| < \varepsilon.$$

The number of hidden nodes in the hidden layer depends on the function we want to approximate. The above property is called the universal approximation property and has both positive as negative implications. On the one hand, this allows us to learn and model various complicated distributions. On the other hand we have to be careful with this power not to overtrain our network and hereby loosing all the generalizing power.

6.3.1 Overtraining

We speak of overtraining when a network *memorizes* the data set by perfectly approximating the target variable for only those input combinations that are found in the data set. This, most of the time, introduces large weights and bad predictions for input combinations other than those found in the data set, as will be explained below.

Figure 6.3 shows the graph of two multilayer perceptrons trained on a data set where $\underline{x}$ is uniform between -5 and 5 and $\underline{y} \sim \mathcal{N}(\sin(\underline{x}), 0.04)$. Both networks had the same structure: two hidden layers with each five nodes and the hyperbolic tangent transfer function. The output node used a linear transfer function. One neural network has been trained by optimizing only the sum-of-squares cost function (the full line), using the scaled conjugate gradient algorithm (see Section 6.6.2). The parameter vector for this network had an L_2-norm of 22.3. The other has been trained using an additional Gaussian prior distribution with a standard deviation of 0.5 for each weight. This is the same as using a *weight decay* regularization term to restrict the parameters from becoming too large (the dashed line). This network had an L_2-norm of only 3.62.

Possible causes of overtraining are an inproper network structure, not enough data points, or an inadequate or absent prior distribution for the network parameters.

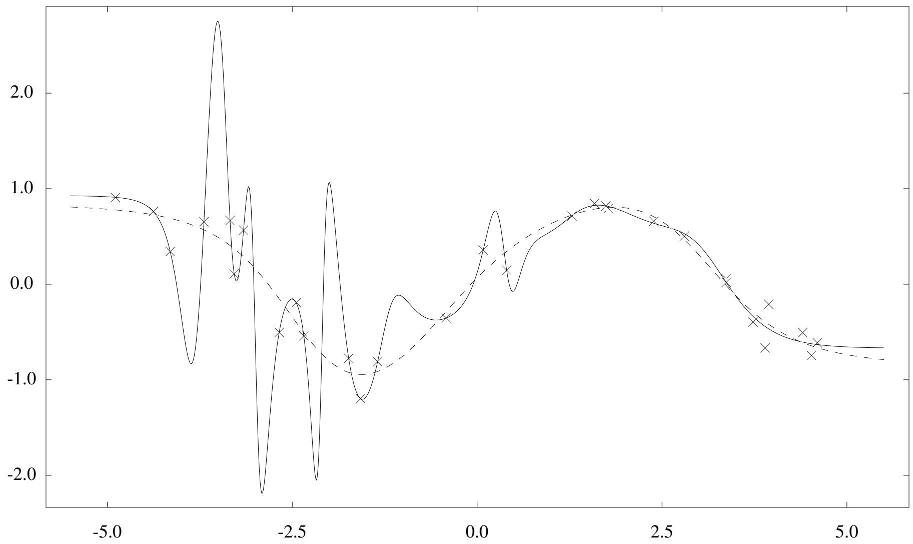


Figure 6.3: The overtrained function (full line) comes from an overtrained multilayer perceptron (22.3 L_2-norm of the weight vector), the dashed function has been learned using a weight decay prior (3.62 L_2-norm of the weight vector).

A network that tries to memorize each input-output pattern, needs the flexibility to change its output considerably based on only a small change of the input pattern. Suppose two input patterns $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are close to each other while their corresponding output values y_1 and y_2 are quite different because of the inherent variability of the data. The relatively small derivative of the transfer function is insufficient to produce the output difference based on the difference between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ if the incoming and outgoing weights of the neuron have a magnitude around one. By multiplying all these weights with a large factor λ, we are in fact using the same unit range weights as before, in combination with a transfer function $f_\lambda(x) = \phi(\lambda x)$. $f_\lambda(.)$ converges pointwise to the sign function, gradually giving the neuron the power to approximate y_1 and y_2 based on the small input difference. The previous reasoning is visualized in Figure 6.4. The gradient of the transfer function $\phi(5.4\,x)$ is large enough to model the output difference between y_1 and y_2 based on the small difference between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

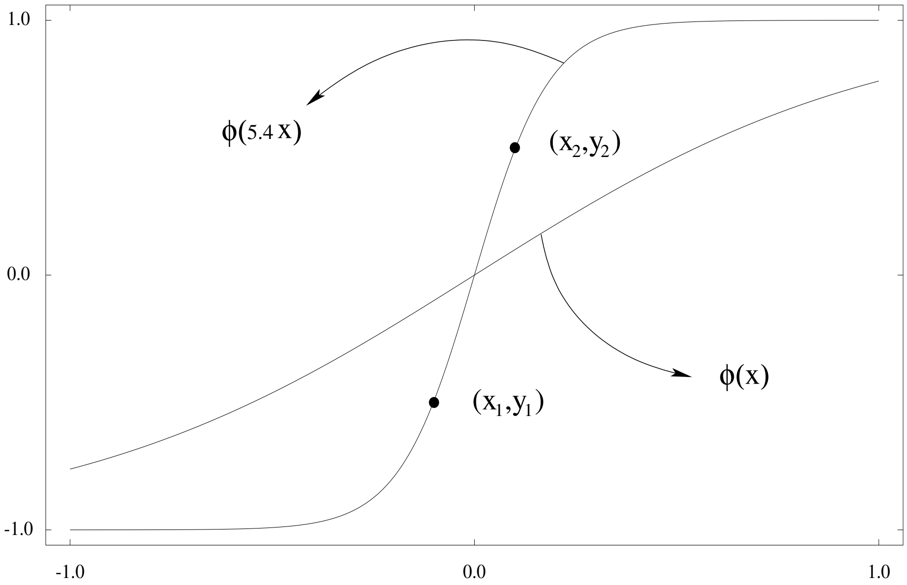


Figure 6.4: Two data points are presented with input patterns $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ close to each other. The transfer function $\phi(x)$ does not have the power to model the outputs, while the transfer function $\phi(5.4\,x)$ has a derivative which is large enough to approximate y_1 and y_2 exactly. Multilayer perceptrons with large weights will therefore indicate overtraining.

This is responsible for the network picking up large weights when it tries to approximate the input patterns exactly. Because of these large weights, the network function becomes very “bumpy” and hereby makes bad predictions for unseen input patterns. We can prevent this behaviour basically using two techniques. One method is based on restricting the network structure so that it does not have enough flexibility (number of neurons) to learn each input pattern exactly. The network will be forced to make compromises, hopefully resulting in a network function with generalization capabilities. The other method tries to find a suitable mapping by restricting the weights from becoming too large.

6.3.2 Early stopping

A first strategy to prevent the network weights from becoming too large is based on dividing the data set into a *training* and a *test* set. The network parameters are initialized near the origin and iteratively updated using some stepwise optimization routine. This optimization routine tries to find the minimum of the error function using only the samples in the training set by taking small steps in the weight space (e.g., steepest descent). The multilayer perceptron will typically need more flexibility than is present close to the origin, and will start wandering towards larger weights. While the optimization process progresses, the norm of the weight vector will grow. The error on the training set will gradually diminish until all the input patterns are approximated exactly. The error on the test set will initially also decrease, indicating that the network function can be generalized to patterns other than in the training set. When the optimization process starts to overtrain the network, the error on the test set will start to grow, indicating a worse performance on samples other than the training samples.

The early stopping technique will stop the optimization process when the test error starts to increase. This method is known to work [77], but has the disadvantage that not all data samples can be used for learning in combination with a restriction on the optimization procedures that can be used: only those procedures that update the weight vector with small steps can be used, preventing the usage of more elaborate methods than gradient descent based methods. This procedure is also hard to interpret from probabilistic point of view in terms of prior and posterior distributions for the network parameters.

6.3.3 Weight decay and a prior distribution

A second strategy uses a penalization or regularization term to penalize the network for using large weights. A common method to achieve this is called *weight decay* and consists in adding the sum of squared weights to the error function

$$\mathrm{Error}^*(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) = \mathrm{Error}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) + \frac{1}{2\sigma^2_{\mathrm{wd}}}\sum_i \boldsymbol{\omega}_i^2. \tag{6.4}$$

We indicate with $\mathrm{Error}^*(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)$ the error term $\mathrm{Error}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)$, *adjusted* with the regularization term.

Where the error function was defined based on the logarithm of the likelihood, this altered error function is based on the logarithm of the posterior distribution of the weights, given the data set $\mathbf{D}$

$$\begin{aligned}
\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) &= \frac{\mathrm{p}(\,\mathbf{D}\,|\,\boldsymbol{\omega}\,)\,\mathrm{p}(\boldsymbol{\omega})}{\mathrm{p}(\mathbf{D})} \\
&\propto \mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)\,\mathrm{p}(\boldsymbol{\omega}) \\
\mathrm{Error}^*(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) &= -\log(\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)) \\
&= \mathrm{Error}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) - \log(\mathrm{p}(\boldsymbol{\omega})) + \mathrm{C}^{\underline{\mathrm{st}}}.
\end{aligned}$$

Since the term coming from $\mathrm{p}(\mathbf{D})$ is independent of the network parameters, it can be dropped. $\mathrm{p}(\boldsymbol{\omega})$ is called the prior distribution for the network parameters. The assumption that each weight $\boldsymbol{\omega}_i$ has a Gaussian prior distribution with zero mean and σ^2_{wd} variance results in the sum of squared weights regularization term in Equation 6.4. The σ_{wd} parameter controls the freedom that is given to the network to use large weights.

Our decision to use only odd transfer functions like the hyperbolic tangent ($\phi(-x) = -\phi(x)$) is related to this Gaussian prior. As will be explained in Section 6.9.1, a network can have different parametrizations resulting in exactly the same input-output mapping: if the signs of all the weights connected to a neuron in a hidden layer are reversed, the overall neural network function will stay the same

$$\ldots \boldsymbol{\omega}_j\, \phi(\sum_i \boldsymbol{\omega}_i\, y_i + b) \ldots = \ldots (-\boldsymbol{\omega}_j)\, \phi(\sum_i (-\boldsymbol{\omega}_i)\, y_i - b) \ldots$$

The sigmoidal transfer function gives rise to a similar symmetry through the property $\mathrm{sigmoid}(x) = 1 - \mathrm{sigmoid}(-x)$

$$\begin{aligned} \ldots b_j &+ \boldsymbol{\omega}_j\, \mathrm{sigmoid}(\sum_i \boldsymbol{\omega}_i\, y_i + b) \ldots \\ &= \ldots b_j + \boldsymbol{\omega}_j\, (1 - \mathrm{sigmoid}(\sum_i (-\boldsymbol{\omega}_i)\, y_i - b)) \ldots \\ &= \ldots (b_j + \boldsymbol{\omega}_j) + (-\boldsymbol{\omega}_j)\, \mathrm{sigmoid}(\sum_i (-\boldsymbol{\omega}_i)\, y_i - b)) \ldots \end{aligned}$$

Although the network mapping stays the same under this transformation, the bias b_j has changed $\boldsymbol{\omega}_j$ in magnitude. This will change the value of the sum of squared weights, resulting in two equivalent network parametrizations getting different probabilities from the posterior distribution $\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)$. This undesired behaviour is our main motivation for using only odd transfer functions like the hyperbolic tangent function.

6.4 Posterior distribution

We will choose for the Bayesian view on regularization, which consists of defining a complexity-based prior distribution which is updated to a posterior distribution with the usage of the data likelihood, based on its nice probabilistic interpretation and its connection with penalization terms in the classical setting:

$$\begin{aligned} \mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) &= \frac{\mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)\,\mathrm{p}(\boldsymbol{\omega})}{\mathrm{p}(\mathbf{D})} \\ &\propto \mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)\,\mathrm{p}(\boldsymbol{\omega}) \\ \mathrm{Error}^*(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) &= -\log(\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)) \\ &= \mathrm{Error}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) + \frac{1}{2\sigma^2_{\mathrm{wd}}} \sum_i \boldsymbol{\omega}_i^2 + \mathrm{C}^{\underline{\mathrm{st}}}. \end{aligned}$$

This posterior distribution is, depending on the actual data set, very complex with many local maxima or near local maxima. Figure 6.5 shows this landscape for an artificial data set. We cannot visualize a multi-dimensional distribution with more than two dimensions. Therefore, we kept all the neural network weights constant and modified only one weight connecting the input to a hidden node, and one weight connecting that same hidden node to the output node. Even this two dimensional slice of the higher dimensional distribution contains two maxima and a saddle point.

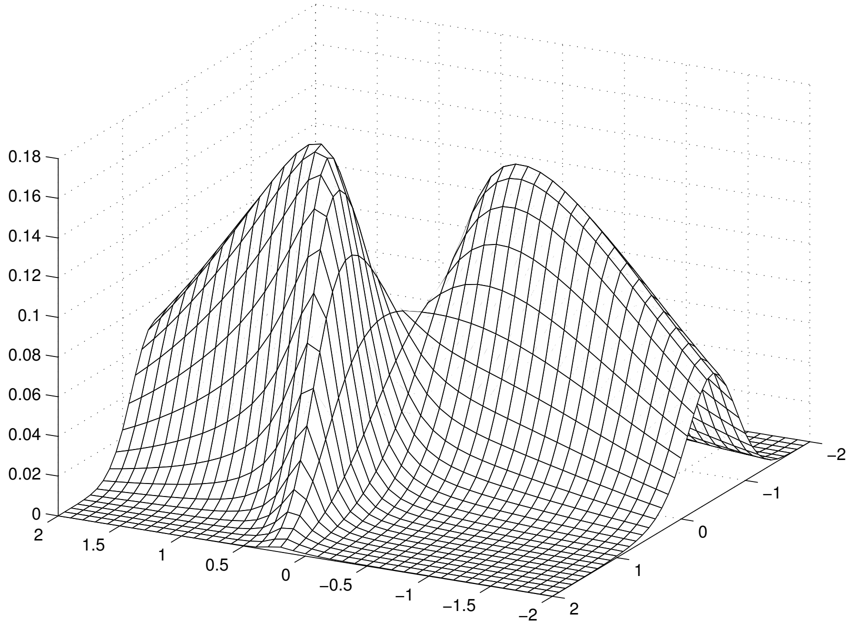


Figure 6.5: The posterior distribution is very complex. We visualized a two-dimensional slice of a higher dimensional distribution by keeping all the weights constant, except one weight connecting the input node to a hidden node and another weight connecting that same hidden node to the output node.

Formulas for the *maximum a posteriori* parametrization cannot be given in closed form, neither can we give a straightforward procedure to sample from this distribution. This problem will be discussed in Section 6.6 and 6.7.

6.5 Pre-processing

As with most probabilistic models, we have to pre-process our data before we can start to learn its underlying distribution. This consists of transforming the variables and constructing features which facilitate the learning of the network. How these features are constructed depend strongly on the type of problem we are trying to solve and can range from simple input selection over principal component analysis to highly problem specific pre-processing. Because we are working with medical measurements, we will restrict the feature extraction phase to selecting the relevant medical variables.

6.5.1 Continuous variables

A multilayer perceptron works most naturally with continuous variables that take values in $\mathbb{R}$. Although the weights from the inputs to the first layer are specifically meant for rescaling the variables, we will standardize each continuous variable to zero mean and unit variance. If we skip this step, two variables equal up to a scaling factor will not be treated the same way; the variable with larger variance will be preferred by our regularization prior because it needs smaller weights to achieve the same effect. Before we standardize our variable, we will check its distribution and its relation with the output variable to decide on a possible transformation that has to be applied: if the output variable were continuous, it would have been natural to look at the scatter plot. Unfortunately, our output variable *Pathology* is discrete, which makes scatter plots less interesting. Therefore we follow the logistic regression approach where a linear combination of the covariates predicts the logarithm of the odds of the response variable. We will estimate these log-odds and use them instead of the class labels to create the so-called logit plots.

The odds of a probability p is defined as $\frac{\mathrm{p}}{1-\mathrm{p}}$. The symbol p indicates the probability of $\mathcal{C}_\mathrm{P}$ and can be estimated by dividing a variable into a few number of bins and using the empirical estimate of p for these bins. These logit plots can suggest a nonlinear transformation of the covariate.

Figure 6.6 shows such a logit plot of a variable before the logarithmic transformation was applied, while Figure 6.7 shows the logit plot after the logarithmic transformation. The logarithm clearly makes the relation between the variable and the log-odds linear, which was our goal.

6.5.2 Ordinal and nominal discrete variables

The second type of variables are discrete variables. They can take only a finite number of different values which are typically in $\mathbb{N}$ or $\mathbb{Z}$. If there is a meaningful order between the values of such a variable, we call them *ordinal* variables and can represent them with only one input node. Instead of normalizing these variables, we will transform them in a linear way so that their values are equally distributed between -1 and 1. An example of such a variable is the age of a person.

If such an ordering is absent, we call them *nominal* variables. Although we represent their values also with natural numbers, these values have no numerical meaning or significance, there is no particular reason why to prefer one possible representation above another. An example of such a variable could be the brand of car that a person drives. Suppose that this variable $\underline{x}$ can take d possible values $\{x_1, \ldots, x_d\}$. The standard way to represent such a variable, is by introducing d *design* or *indicator* variables I_{x_i}, each of which takes the value 1 if $\underline{x} = x_i$ and 0 otherwise. Although such a variable in a linear or logistic regression problem is often represented with $d-1$ variables by connecting the event "$\underline{x} = x_d$" to "all indicator variables equal to zero", we will only do so for binary variables. If we use d design variables to represent $\underline{x}$, we can permutate the meaning of the values of $\underline{x}$ and still the resulting equivalent network parametrization will have the same weight decay value. If we use a $d-1$ coding scheme, this will in general not be the case anymore.

Some variables are the combination of a continuous variable and one or more design variables. In the ovarian tumour problem, we have variables like *PSV* which make only sense if there is blood flow through the tumour (*Colour Score* $\neq$ *1*). For a patient with blood flow in the tumour, this variable will indicate the peak systolic velocity of this blood flow. If there is no such blood flow, *PSV* is not relevant and will be set to zero. We will introduce a design variable to indicate if a certain variable is relevant or not. Figures 6.6 and 6.7 show such a variable. The special value "not relevant" is represented with a square.

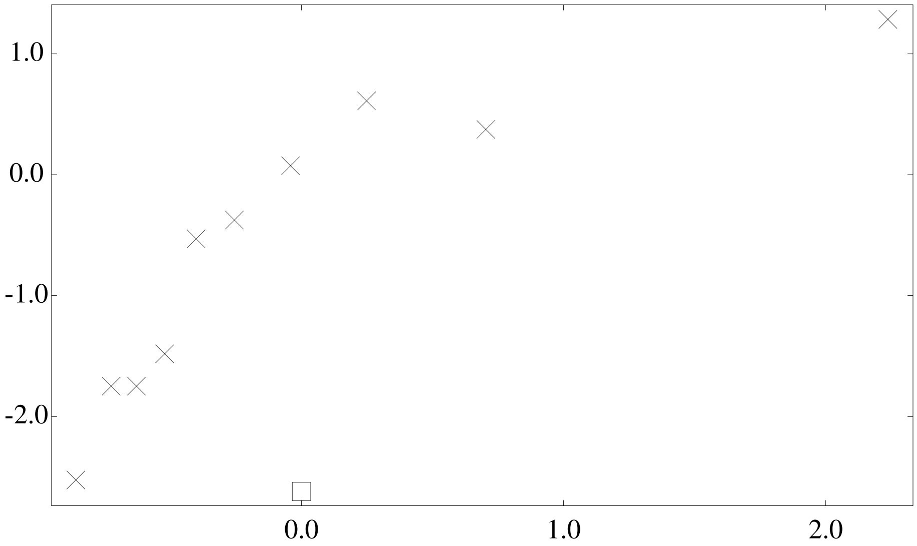


Figure 6.6: The log odds scatter plot for the *PSV* variable, before a logarithmic transformation. The special value "not relevant" is dealt with by introducing a design variable and is indicated by the square. The crosses indicate the log odds of the bins with equal number of samples inside.

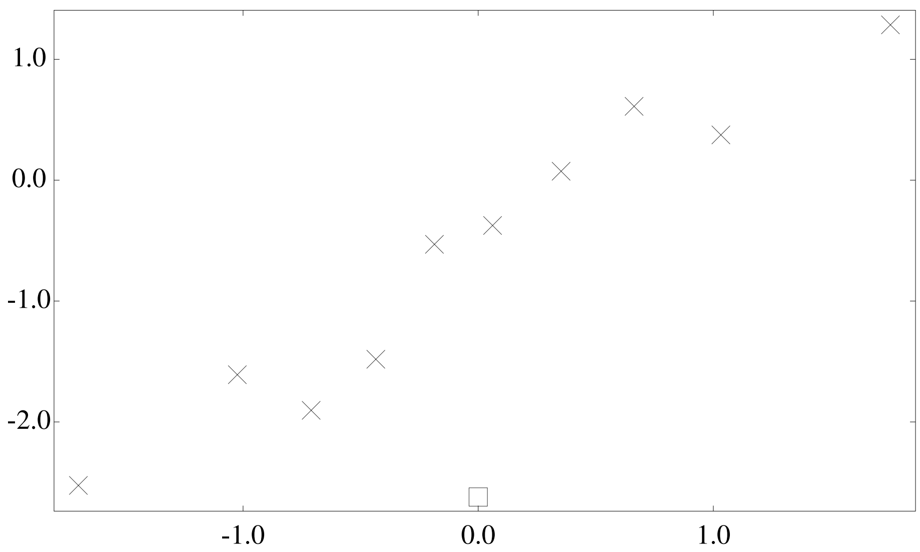


Figure 6.7: The log odds scatter plot for the *PSV* variable, after a logarithmic transformation and standardization. The special value “not relevant” is dealt with by introducing a design variable and is indicated by the square. The crosses indicate the log odds of the bins with equal number of samples inside.

6.5.3 Input selection

As indicated previously, we will select a subset of the variables as inputs for the multilayer perceptron. Ideal would be to divide the data set in a training and test set and compare the performance of models with different input variables by learning them using the training set and computing their performance based on the test set. The computational complexity arising from this setup easily becomes too much because of the combinatorial number of different input sets that have to be checked, requiring an optimization in each step. A possible strategy to ease the computational needs is to select an input set based on a linear or logistic regression model and use this for the neural network model. Additional stepwise input selection procedures can refine this initial set.

6.6 Optimization

Once we have pre-processed our data set and selected the relevant variables, we can start learning our neural network model. In the classical framework, as opposed to the Bayesian framework, we are looking for *one* optimal parametrization $\boldsymbol{\omega}^*$, the weight vector which has the highest posterior density. Using $\boldsymbol{\omega}^*$, we can make a prediction for a new data sample $\boldsymbol{x}$ with the neural network function $\mathrm{mlp}(\boldsymbol{x} \,|\, \boldsymbol{\omega}^*)$. As explained in Section 6.2, this function represents either a regression mean or a class membership probability.

Because the logarithmic function is strictly monotone, finding the maximum a posteriori parametrization is equivalent to the minimum error weight vector $\boldsymbol{\omega}^* = \operatorname{argmin}_{\boldsymbol{\omega}}(\text{Error}^*(\boldsymbol{\omega} \,|\, \mathbf{D}))$. Note that the regularization term in $\text{Error}^*(\boldsymbol{\omega} \,|\, \mathbf{D})$ makes sure that there exists at least one global minimum of the error function. To mark the difference with the Bayesian framework, we will speak about error functions in the classical framework instead of posterior distributions.

Since this error landscape is complex and no closed form solution exists, we have to resort to minimization procedures to find $\boldsymbol{\omega}^*$. These minimization procedures come in different flavours, requirements and capabilities.

6.6.1 Steepest descent

A first method is called *steepest descent*. This procedure starts at some random weight vector $\boldsymbol{\omega}_0$ and takes small steps in the negative direction of the gradient:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta \nabla \, \text{Error}^*(\boldsymbol{\omega}_t \,|\, \mathbf{D}).$$

We are simulating a particle in weight space where the speed at each step is given by $-\nabla \text{Error}^*(\boldsymbol{\omega} \,|\, \mathbf{D})$.

This method has been well studied and is usable, but has some serious drawbacks. First of all, the step size η has to be chosen carefully. If it is too small, only very small steps will be taken resulting in a huge number of steps that have to be taken before convergence is reached. If it is too large, we can overshoot a minimum or end up in oscillatory behaviour. This procedure is sensitive to local minima and cannot get out of such a minimum if it enters one.

In reaction to these drawbacks, several improvements have been proposed. They include taking into account a *momentum term* during the optimization

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta \nabla \, \text{Error}^*(\boldsymbol{\omega}_t \,|\, \mathbf{D}) + \mu \, \boldsymbol{\omega}_t.$$

We are now simulating a *mass* that moves through weight space. The inertia of this mass is responsible for accumulating speed or spending this accumulated speed to escape a local minimum.

Another procedure, called *resilient learning*, respectively increases or decreases the step size for each weight *separately* based on whether the sign of the corresponding last two gradient values was the same or not.

6.6.2 Scaled conjugate gradient

Another class of function minimizers approximate the error function with a quadratic Taylor series

$$\begin{aligned}\mathrm{Error}(\boldsymbol{\omega}+\boldsymbol{s}) &= \mathrm{Error}(\boldsymbol{\omega}) + \sum_i \frac{\partial \mathrm{Error}}{\partial \boldsymbol{\omega}_i} s_i + \frac{1}{2}\sum_{i,j}\frac{\partial^2 \mathrm{Error}}{\partial \boldsymbol{\omega}_i \partial \boldsymbol{\omega}_j} s_i s_j + \ldots \\ &\approx a + \boldsymbol{b}^T \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T \boldsymbol{H}\boldsymbol{s} \\ &= \mathrm{Error}_{\boldsymbol{\omega}}^{\sim}(\boldsymbol{s}),\end{aligned}$$

where

$$a = \mathrm{Error}(\boldsymbol{\omega}) \quad \boldsymbol{b} = \nabla \mathrm{Error}(\boldsymbol{\omega}) \quad \boldsymbol{H}_{i,j} = \frac{\partial^2 \mathrm{Error}}{\partial \boldsymbol{\omega}_i \partial \boldsymbol{\omega}_j}(\boldsymbol{\omega}).$$

Using this approximation of our error function, we take that step $\boldsymbol{s}^*$ where $\mathrm{Error}_{\boldsymbol{\omega}}^{\sim}(\cdot)$ has its minimum

$$\nabla \mathrm{Error}_{\boldsymbol{\omega}}^{\sim}(\boldsymbol{s}^*) = \boldsymbol{H}\boldsymbol{s}^* + \boldsymbol{b} = \boldsymbol{0}.$$

Finding $\boldsymbol{s}^*$ involves solving a linear system of equations specified by the Hessian matrix $\boldsymbol{H}$ and the gradient vector $\nabla \mathrm{Error}(\boldsymbol{\omega})$. One option is to solve this system using the singular value decomposition of $\boldsymbol{H}$. This requires that we know $\boldsymbol{H}$ and can store it.

Another strategy is based on finding *conjugate* directions $\boldsymbol{q}_i$ and finding the minimum of $\mathrm{Error}_{\boldsymbol{\omega}}^{\sim}(\cdot)$ along those directions. Suppose that $\boldsymbol{p}_1 = \alpha_1 \boldsymbol{q}_1$ minimizes $\mathrm{Error}_{\boldsymbol{\omega}}^{\sim}(\cdot)$ along the direction $\boldsymbol{q}_1$. This means that the projection of the gradient in $\boldsymbol{p}_1$ on $\boldsymbol{q}_1$ is zero:

$$\boldsymbol{q}_1^T \nabla \mathrm{Error}_{\boldsymbol{\omega}}^{\sim}(\boldsymbol{p}_1) = \boldsymbol{q}_1^T \boldsymbol{b} + \alpha_1 \boldsymbol{q}_1^T \boldsymbol{H}\boldsymbol{q}_1 = \boldsymbol{0}.$$

Next, we minimize $\mathrm{Error}_{\boldsymbol{\omega}}^{\sim}(\cdot)$ along another direction $\boldsymbol{q}_2$. This requires our gradient in $\boldsymbol{p}_2 = \alpha_2 \boldsymbol{q}_2$ to be again perpendicular to $\boldsymbol{q}_2$. But we are also careful not to destroy the partial minimum we found in $\boldsymbol{p}_1$; we require the gradient in $\boldsymbol{p}_2$ to be similarly perpendicular to $\boldsymbol{q}_1$:

$$\begin{aligned}\boldsymbol{q}_1^T \nabla \mathrm{Error}_{\boldsymbol{\omega}}^{\sim}(\boldsymbol{p}_1+\boldsymbol{p}_2) &= \boldsymbol{q}_1^T \boldsymbol{b} + \alpha_1 \boldsymbol{q}_1^T \boldsymbol{H}\boldsymbol{q}_1 + \alpha_2 \boldsymbol{q}_1^T \boldsymbol{H}\boldsymbol{q}_2 \\ &= \alpha_2 \boldsymbol{q}_1^T \boldsymbol{H}\boldsymbol{q}_2 \\ &= 0.\end{aligned}$$

This requires our directions $\boldsymbol{q}_i$ to be *conjugate* with respect to the Hessian matrix:

$$\boldsymbol{q}_i^T \boldsymbol{H}\boldsymbol{q}_j = 0, \;\; i \neq j.$$

We can find these conjugate directions — without having to compute $\boldsymbol{H}$ — by minimizing $\mathrm{Error}_{\boldsymbol{\omega}}^{\sim}(\cdot)$ along some directions with a one-dimensional minimization routine, also known as a line search. Although this works nicely, such a line search generally takes twelve function evaluations or gradient computations to find the appropriate α_i along a direction $\boldsymbol{q}_i$.

Another approach [61] uses second order information along the direction $\boldsymbol{q}_i$ to directly find the minimum. This second derivative can be computed numerically with only two gradient computations, which is most of the time faster than performing a line search. No entire Hessian matrix has to be stored, leaving the memory complexity unaffected.

These procedures generally work very bad: far from the global minimum, our approximation will have an indefinite Hessian matrix, and thus no minimum. This problem is solved by making the Hessian matrix positive definite using a Levenberg-Marquardt approach, and modifying the steps that will be taken.

If the Hessian had to be adjusted a lot to become positive definite, smaller steps will be taken, shifting the effective behaviour of the algorithm towards steepest descent.

6.7 Bayesian simulation

Instead of finding one optimal parametrization $\boldsymbol{\omega}^*$ — the maximum a posteriori parametrization — to make a prediction for a new subject $\boldsymbol{x}$, the Bayesian approach uses the posterior distribution over the neural network weights to make this prediction.

To make things clear, let us assume that we want to perform a classification task and thus want to predict the class label $\underline{t}$ of a certain subject based on the input vector $\boldsymbol{x}$. We have also access to a supervised data set $\mathbf{D}$, which contains a number of records with known classification labels, some background information ξ, and assume that we can model the classification problem with a multilayer perceptron. We can write out the classification probability as

$$\begin{aligned} \mathrm{P}(\,\underline{t} = \mathcal{C}_{\mathrm{P}} \,|\, \boldsymbol{x}, \mathbf{D}, \xi\,) &= \int_{\boldsymbol{\Omega}} \mathrm{P}(\,\underline{t} = \mathcal{C}_{\mathrm{P}} \,|\, \boldsymbol{x}, \mathbf{D}\,)\, \mathrm{p}(\,\boldsymbol{\omega} \,|\, \mathbf{D}, \xi\,)\, d\boldsymbol{\omega} \\ &= \int_{\boldsymbol{\Omega}} \mathrm{mlp}(\,\boldsymbol{x} \,|\, \boldsymbol{\omega}\,)\, \mathrm{p}(\,\boldsymbol{\omega} \,|\, \mathbf{D}, \xi\,)\, d\boldsymbol{\omega} \end{aligned} \tag{6.5}$$

From the above formula, we see that the Bayesian framework uses *all* multilayer perceptron parametrizations, and weights each parametrization according to the posterior distribution $\mathrm{p}(\,\boldsymbol{\omega} \,|\, \mathbf{D}, \xi\,)$.

If we can compute this integral, we know the probability that the subject belongs to class $\mathcal{C}_{\mathrm{P}}$ given the observation vector $\boldsymbol{x}$ and the data set $\mathbf{D}$. Unfortunately, it will not be straightforward to compute this integral. Since we cannot compute this integral analytically, we will approximate it using a Monte Carlo summation, as will be explained hereunder.

In Section 5.3.1 we will take a closer look at $\mathrm{p}(\,\boldsymbol{\omega} \,|\, \mathbf{D}, \xi\,) \propto \mathcal{L}(\,\boldsymbol{\omega} \,|\, \mathbf{D}\,)\, \mathrm{p}(\,\boldsymbol{\omega} \,|\, \xi\,)$. The likelihood part is fixed, but we have several options to specify the prior distribution $\mathrm{p}(\,\boldsymbol{\omega} \,|\, \xi\,)$, depending on what background knowledge ξ we have. This can be either a complexity-based prior or an informative prior, which could be defined using the transformation technique introduced in Section 4. Because this transformation technique provides us *only* with a procedure to sample from the informative prior $\mathrm{p}(\,\boldsymbol{\omega}\,|\,\xi\,)$, we will estimate this distribution using some parametric distribution. In Section 6.8, we introduce a range of distributions with an increasing expressive power that could be used for this.

Section 6.9 finally discusses the technical details when estimating a distribution in neural network weightspace successfully, dealing with the symmetries that may occur.

Let us first concentrate on Equation 6.5. It is not possible to compute this integral exactly, leaving us with approximation techniques. A Monte Carlo summation [37] approximates an integral over the support $\mathcal{S}$ of a density function $\mathrm{p}(\cdot)$:

$$\begin{aligned}\int_{\mathcal{S}} f(\boldsymbol{x})\,\mathrm{p}(\boldsymbol{x})\,d\boldsymbol{x} &\approx \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i) \quad \text{where } \boldsymbol{x}_i \sim p(\boldsymbol{x}) \\ &= \mathrm{MC}_N^{\sim}(f(\cdot))\end{aligned}$$

The name “Monte Carlo” comes from the gambling place Monte Carlo, referring to the laws of chance that are involved and finds unfortunately its first application in building better atomic bombs by Ulam and von Neumann.

The variance of this Monte Carlo estimator, when the vectors $\boldsymbol{\omega}_i$ are independent, is

$$\mathrm{V}[\mathrm{MC}_N^{\sim}(f(\cdot))] = \frac{1}{N}\mathrm{V}[f(\underline{\boldsymbol{\omega}})] = \frac{1}{N}\int_{\boldsymbol{\Omega}} (f(\boldsymbol{\omega}) - \mathrm{E}[f(\underline{\boldsymbol{\omega}})])^2 \mathrm{p}(\boldsymbol{\omega})\,d\boldsymbol{\omega},$$

which means that the standard deviation has a rather slow convergence rate of $1/\sqrt{N}$. Because there is not much we can do to improve this rate, we should try to make the second factor $\int_{\boldsymbol{\Omega}}(f(\boldsymbol{\omega}) - \mathrm{E}[f(\underline{\boldsymbol{\omega}})])^2\mathrm{p}(\boldsymbol{\omega})\,d\boldsymbol{\omega}$ as small as possible. This term measures the variance of the function $f(\cdot)$ under the distribution $\mathrm{p}(\cdot)$. A common approach is to rewrite the integrand $f(\boldsymbol{\omega})\,\mathrm{p}(\boldsymbol{\omega})$ as $f^*(\boldsymbol{\omega})\,\mathrm{p}^*(\boldsymbol{\omega})$ in such a way that $f^*(\cdot)$ becomes as constant as possible while we can still generate random numbers from $\mathrm{p}^*(\cdot)$ in an efficient way.

In our case, we can determine immediately two possible distributions we can use to create a Monte Carlo summation

$$\begin{aligned}f(\boldsymbol{\omega})\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) &= f(\boldsymbol{\omega})\frac{\mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)}{\mathrm{p}(\mathbf{D})}\mathrm{p}(\boldsymbol{\omega}) \\ &= f^*(\boldsymbol{\omega})\,\mathrm{p}(\boldsymbol{\omega}).\end{aligned}$$

We can either use the posterior or the prior distribution in the Monte Carlo summation.

6.7.1 Weighted prior samples and rejection sampling

The most straightforward method uses the a priori distribution based Monte Carlo expansion

$$\int_{\boldsymbol{\Omega}} f(\boldsymbol{\omega})\frac{\mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)}{\mathrm{p}(\mathbf{D})}\,\mathrm{p}(\boldsymbol{\omega})\,d\boldsymbol{\omega} \approx \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{\omega}_i)\,\frac{\mathcal{L}(\,\boldsymbol{\omega}_i\,|\,\mathbf{D}\,)}{\mathrm{p}(\mathbf{D})} \quad \text{with } \boldsymbol{\omega}_i \sim \mathrm{p}(\boldsymbol{\omega}).$$

This procedure has the benefit that independent parametrizations $\boldsymbol{\omega}_i$ can be generated in an efficient manner from the prior distribution and results in a summation where the values $f(\boldsymbol{\omega}_i)$ of the function in question are reweighted with the likelihood of that weight vector given the data. It is this likelihood factor that results in a high variance for the new function of interest $f^*(\boldsymbol{\omega}) = f(\boldsymbol{\omega})\,\mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)/\mathrm{p}(\mathbf{D})$ under the prior distribution $\mathrm{p}(\cdot)$. It can take a long time before we, by accident, draw a parametrization $\boldsymbol{\omega}_i$ from $\mathrm{p}(\boldsymbol{\omega})$ that has a non-zero likelihood value. This renders the prior Monte Carlo method virtually impossible to use. To make things even worse, we have to estimate the factor

$$\mathrm{p}(\mathbf{D}) = \int_{\boldsymbol{\Omega}} \mathrm{p}(\,\mathbf{D}\,|\,\boldsymbol{\omega}\,)\,\mathrm{p}(\boldsymbol{\omega})\,d\boldsymbol{\omega}$$

in a similar manner.

The solution is to use the expansion with respect to the posterior distribution

$$\int_{\boldsymbol{\Omega}} f(\boldsymbol{\omega})\,\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)\,d\boldsymbol{\omega} \approx \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{\omega}_i) \quad \text{with } \boldsymbol{\omega}_i \sim \mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,).$$

It is reasonable to expect the function of interest, $f(\cdot)$, to have small variance with respect to the posterior distribution, which will result in an acceptable variance for our Monte Carlo estimator. On the other hand, it is much harder this time to generate random vectors from the posterior distribution. An analytic form of this distribution is not given, which leaves the direct generation of random vectors impossible.

A possible strategy to generate from $\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) \propto \mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)\,\mathrm{p}(\boldsymbol{\omega})$, is to use *rejection sampling* [74] using the prior distribution:

Initialization: Determine $\max_{\boldsymbol{\omega}} \mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)$ or an upper bound.

1. Generate $\boldsymbol{\omega}$ according to $\mathrm{p}(\cdot)$.
2. Accept $\boldsymbol{\omega}$ with probability $\frac{\mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)}{\max_{\boldsymbol{\omega}} \mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)}$, else return to Step 1.

This procedure samples from a distribution

$$\mathrm{p}_{\text{rejection}}(\boldsymbol{\omega}) \propto \mathrm{p}(\boldsymbol{\omega})\frac{\mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)}{\max_{\boldsymbol{\omega}} \mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)},$$

which has to be equal to the posterior distribution since each distribution integrates to one.

Although rejection sampling is a nice method, its application here is equally naive as the prior expansion. The problem is again the term $\mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)$, which is responsible for the huge difference between $\mathrm{p}(\boldsymbol{\omega})$ and $\mathrm{p}(\boldsymbol{\omega})\max_{\boldsymbol{\omega}} \mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)$.

To demonstrate the ineffectiveness of this rejection sampling method, we created a small artificial data set containing only 4 data points, and generated five network parametrizations from the posterior distribution (see Figure 6.8). To get these five weight vectors accepted, it took 4 399 397 rejected vectors. The inefficiency of this method even grows as the number of parameters or the number of data points increases.

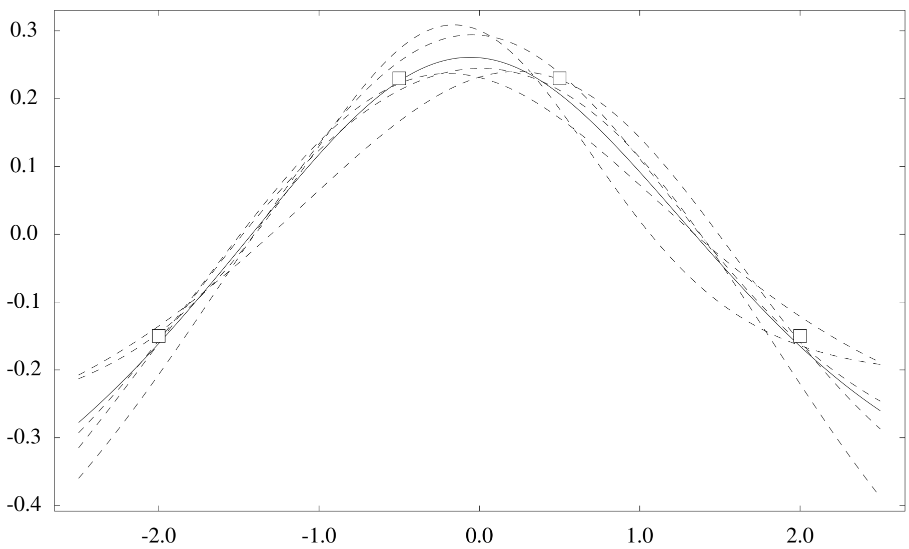


Figure 6.8: Five network functions, together with their Monte Carlo sum, drawn from the a posteriori distribution using the rejection sampling method.

6.7.2 Markov chain methods

When direct sampling from some distribution via rejection sampling or some similar method fails, we can look into Markov chain methods. A Markov chain can be thought of as a machine that jumps from one state to another in discrete time according to specific rules. We represent with ϑ^t the state of our Markov chain at time t. To describe a Markov chain, we need the probability distribution $\mathrm{p}(\vartheta^{t+1} \mid \vartheta^t)$ to arrive in state ϑ^{t+1} at time $t+1$ if we were at ϑ^t at time t. A Markov chain assumes that this distribution is time invariant. Another central assumption called the Markov property is that a Markov chain *forgets*. With forgetting, we mean that the knowledge of the state of the chain at time t is the only thing we need to predict its state at time $t+1$. The additional knowledge of states $\vartheta^{t-1}, \vartheta^{t-2}, \ldots$ does not give any new information:

$$\mathrm{p}(\vartheta^{t+1} \mid \vartheta^t, \vartheta^{t-1}, \vartheta^{t-2}, \ldots) = \mathrm{p}(\vartheta^{t+1} \mid \vartheta^t).$$

If we run a Markov chain for a long time, the distribution of its state will become independent of the initial state. Many Markov chain applications will choose the transition probabilities carefully such that this long run distribution will have special properties.

The methods described here are based on constructing a Markov chain for the weight vector of a neural network. By carefully defining the transition probabilities to end up in $\boldsymbol{\omega}^{t+1}$ from state $\boldsymbol{\omega}^t$, we can determine the equilibrium distribution of our Markov chain, the distribution of the samples if we run the chain long enough. We denote this transition distribution from $\boldsymbol{\omega}^t$ to $\boldsymbol{\omega}^{t+1}$ with $T(\,\boldsymbol{\omega}^{t+1}\,|\,\boldsymbol{\omega}^t\,)$.

An equilibrium distribution $q(\boldsymbol{\omega})$ of a Markov chain with transition distribution $T(\,\boldsymbol{\omega}'\,|\,\boldsymbol{\omega}\,)$ has the property that, if $\boldsymbol{\omega}^t$ has distribution $q(\boldsymbol{\omega})$, all the following parametrizations $\boldsymbol{\omega}^{t+i},\ i > 0$ are also distributed according to $q(\boldsymbol{\omega})$. This property translates to

$$q(\boldsymbol{\omega}') = \int_{\boldsymbol{\Omega}} T(\,\boldsymbol{\omega}'\,|\,\boldsymbol{\omega}\,)\,q(\boldsymbol{\omega})\,d\boldsymbol{\omega}$$

and is certainly matched if

$$T(\,\boldsymbol{\omega}'\,|\,\boldsymbol{\omega}\,)\,q(\boldsymbol{\omega}) = T(\,\boldsymbol{\omega}\,|\,\boldsymbol{\omega}'\,)\,q(\boldsymbol{\omega}') \quad \forall \boldsymbol{\omega}, \boldsymbol{\omega}' \in \boldsymbol{\Omega}. \tag{6.6}$$

A Markov chain is called *irreducible* if every state can be reached from every other state in a finite number of steps. If a Markov chain has an equilibrium distribution $q(\cdot)$ and is irreducible, its equilibrium distribution will be unique and converges to this distribution from every start vector $\boldsymbol{\omega}^1$.

Metropolis Markov Chain

An often used Markov chain is called the Metropolis Markov chain [16]. We want to generate samples from the posterior distribution $\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)$ and define a distribution $S(\,\boldsymbol{\omega}^*\,|\,\boldsymbol{\omega}^t\,)$ that we will use to generate candidate states. The algorithm to generate $\boldsymbol{\omega}^{t+1}$ from $\boldsymbol{\omega}^t$ is represented in Figure 6.9 and goes as follows:

1. Generate a candidate state $\boldsymbol{\omega}^*$ from $S(\,\boldsymbol{\omega}^*\,|\,\boldsymbol{\omega}^t\,)$.
2. If $\mathrm{p}(\,\boldsymbol{\omega}^*\,|\,\mathbf{D}\,) \geq \mathrm{p}(\,\boldsymbol{\omega}^t\,|\,\mathbf{D}\,)$, accept candidate $\boldsymbol{\omega}^*$, else accept candidate $\boldsymbol{\omega}^*$ with probability $\mathrm{p}(\,\boldsymbol{\omega}^*\,|\,\mathbf{D}\,)/\mathrm{p}(\,\boldsymbol{\omega}^t\,|\,\mathbf{D}\,)$.
3. If $\boldsymbol{\omega}^*$ is accepted, return $\boldsymbol{\omega}^{t+1} = \boldsymbol{\omega}^*$, else return $\boldsymbol{\omega}^{t+1} = \boldsymbol{\omega}^t$.

The transition distribution can thus be written as

$$T(\,\boldsymbol{\omega}'\,|\,\boldsymbol{\omega}\,) = S(\,\boldsymbol{\omega}'\,|\,\boldsymbol{\omega}\,)\,\min\left(1, \frac{\mathrm{p}(\,\boldsymbol{\omega}'\,|\,\mathbf{D}\,)}{\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)}\right).$$

If we choose $S(\,\boldsymbol{\omega}'\,|\,\boldsymbol{\omega}\,)$ symmetric ($S(\,\boldsymbol{\omega}'\,|\,\boldsymbol{\omega}\,) = S(\,\boldsymbol{\omega}\,|\,\boldsymbol{\omega}'\,)$), we can prove that $\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)$ is the equilibrium distribution:

$$\begin{aligned}
T(\,\boldsymbol{\omega}'\,|\,\boldsymbol{\omega}\,)\,\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) &= S(\,\boldsymbol{\omega}'\,|\,\boldsymbol{\omega}\,)\,\min\left(1, \frac{\mathrm{p}(\,\boldsymbol{\omega}'\,|\,\mathbf{D}\,)}{\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)}\right)\,\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,) \\
&= S(\,\boldsymbol{\omega}'\,|\,\boldsymbol{\omega}\,)\,\min(\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,), \mathrm{p}(\,\boldsymbol{\omega}'\,|\,\mathbf{D}\,)) \\
&= S(\,\boldsymbol{\omega}\,|\,\boldsymbol{\omega}'\,)\,\min(\mathrm{p}(\,\boldsymbol{\omega}'\,|\,\mathbf{D}\,), \mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)) \\
&= S(\,\boldsymbol{\omega}\,|\,\boldsymbol{\omega}'\,)\,\min\left(1, \frac{\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)}{\mathrm{p}(\,\boldsymbol{\omega}'\,|\,\mathbf{D}\,)}\right)\,\mathrm{p}(\,\boldsymbol{\omega}'\,|\,\mathbf{D}\,) \\
&= T(\,\boldsymbol{\omega}\,|\,\boldsymbol{\omega}'\,)\,\mathrm{p}(\,\boldsymbol{\omega}'\,|\,\mathbf{D}\,),
\end{aligned}$$

and thus Equation 6.6 is fulfilled.

Figure 6.9 presents a schematic representation of the algorithm that is used to generate new states of the Markov chain. If the value of the posterior distribution at $\boldsymbol{\omega}^*$ is higher than in the previous state, we always accept this new state. This behaviour alone would result in an optimization routine. By accepting also states with a lower density, we can escape local minima, something we expect from a Markov chain.

Crucial for this Markov chain is the choice of the *candidate generating* distribution $S(\,\boldsymbol{\omega}^* \,|\, \boldsymbol{\omega}\,)$. We are free to choose $S(\,\boldsymbol{\omega}^* \,|\, \boldsymbol{\omega}\,)$ as long as it is symmetric. A common choice is a multivariate Gaussian distribution with mean $\boldsymbol{\omega}$

$$S(\,\boldsymbol{\omega}^* \,|\, \boldsymbol{\omega}\,) = \mathcal{N}(\boldsymbol{\omega}, \alpha\boldsymbol{\mathcal{I}}).$$

With α, we can control how far away we will look for a candidate parametrization. For some regions of the weight space, we can set this quite large to achieve a reasonable acceptance rate, while for other regions we are practically in a valley and need a small α. Unfortunately, we cannot modify α during the Markov chain as this would destroy the symmetry of $S(\,\boldsymbol{\omega}^* \,|\, \boldsymbol{\omega}\,)$. By observing this, we have to set α relatively small, which will result in small steps and thus a slow walk through the weight space. And even this slow walk is far from optimal since $S(\,\boldsymbol{\omega}^* \,|\, \boldsymbol{\omega}\,)$ does not use any information from the posterior to generate new candidates. This can be a benefit for solving problems where only the posterior distribution can be computed, and for instance no gradient, but here it will render the Markov chain unusable for generating samples from the posterior distribution, especially if a lot of parameters are involved.

Hybrid Monte Carlo Markov Chain

To deal with the problems of the Metropolis method to generate new states for our Markov chain, we will use the *hybrid Monte Carlo Markov chain* method [63]. This method will use higher-order derivatives of the density and use this to speed up the travel through weight space.

The algorithm requires that we write down the distribution we want to sample from in terms of *energy.* The reason is that it originates from physics, where we want to sample according to the Boltzmann distribution of a system. Such a system has some *potential* energy $E(\boldsymbol{q})$, where $\boldsymbol{q}$ is a vector of space coordinates. The Boltzmann distribution (also called the *canonical* distribution) for the variables $\boldsymbol{q}$ is

$$\mathrm{p}_{\mathrm{Boltzmann}}(\boldsymbol{q}) \propto e^{-E(\boldsymbol{q})}.$$

Nothing stops us to define a virtual physical system where we define $\boldsymbol{q}$ as $\boldsymbol{\omega}$ and the potential energy as

$$E(\boldsymbol{\omega}) = -\log(\mathrm{p}(\,\boldsymbol{\omega} \,|\, \mathbf{D}\,)) + \mathrm{C}^{\underline{\mathrm{st}}}.$$

The Boltzmann distribution where the algorithm will sample from is now our wanted posterior distribution.

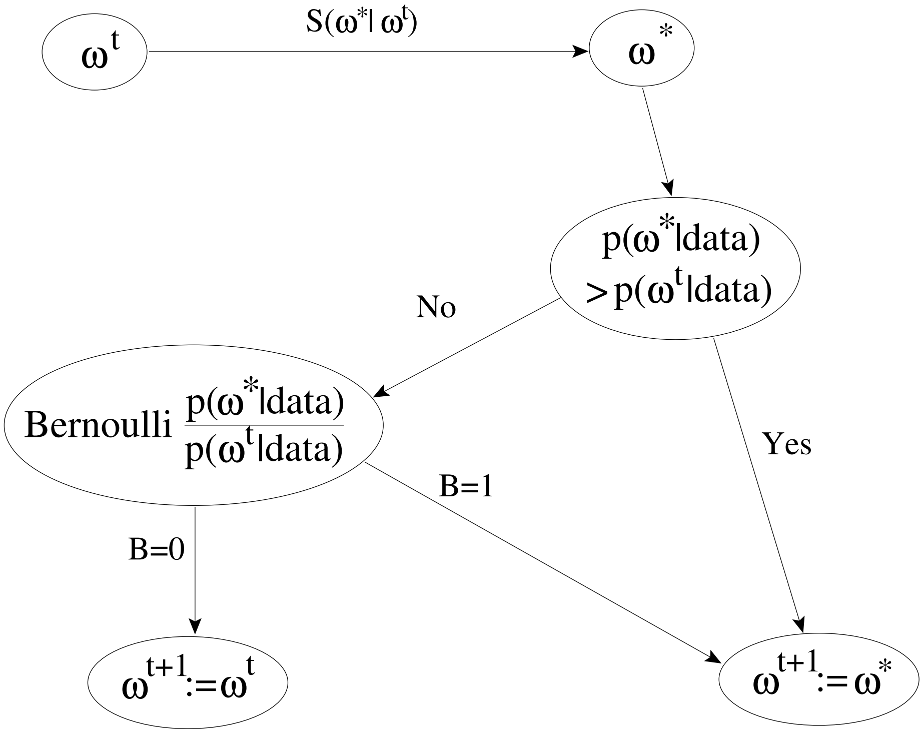

ω^t
$S(\omega^* | \omega^t)$
ω^*
$p(\omega^* | \text{data}) > p(\omega^t | \text{data})$
No
Yes
Bernoulli $\frac{p(\omega^* | \text{data})}{p(\omega^t | \text{data})}$
B=1
B=0
$\omega^{t+1} := \omega^t$
$\omega^{t+1} := \omega^*$

Figure 6.9: A schematic representation of the Metropolis algorithm.

In addition to this potential distribution, we also have to create a vector containing momentum terms $\boldsymbol{p}_i$ for each parameter $\boldsymbol{q}_i$. This momentum vector will determine the kinetic energy of the system as is usually done in physics

$$K(\boldsymbol{p}) = \sum_i \frac{\boldsymbol{p}_i}{2m_i},$$

where m_i is the *mass* of particle i. Because we have no real physical system, we set the masses of the particles to 1. Note that the mass has nothing to do with the name *weight* that is frequently used to indicate the parameters of a neural network. The total energy of the system is now given by the Hamiltonian

$$H(\boldsymbol{\omega}, \boldsymbol{p}) = E(\boldsymbol{\omega}) + K(\boldsymbol{p}),$$

and the canonical distribution for both $\boldsymbol{\omega}$ and $\boldsymbol{p}$ becomes

$$\begin{aligned} \mathrm{p}(\boldsymbol{\omega}, \boldsymbol{p}) &\propto e^{-H(\boldsymbol{\omega}, \boldsymbol{p})} \\ &= e^{-E(\boldsymbol{\omega})} e^{-K(\boldsymbol{p})} \\ &= \mathrm{p}(\, \boldsymbol{\omega} \,|\, \mathbf{D} \,)\, e^{-K(\boldsymbol{p})}. \end{aligned}$$

The variables $\boldsymbol{\omega}$ and $\boldsymbol{p}$ are clearly independent and we want to sample from the marginal distribution for $\boldsymbol{\omega}$. The idea is to construct a Markov chain that converges to the Boltzmann distribution for both $\boldsymbol{\omega}$ and $\boldsymbol{p}$, and only retain the values for $\boldsymbol{\omega}$. To create the Markov chain, we will make use of the Hamiltonian dynamics

$$\begin{aligned}\frac{d\boldsymbol{\omega}_i}{d\tau} &= +\frac{\partial H}{\partial \boldsymbol{p}_i} = \frac{\boldsymbol{p}_i}{m_i}\\ \frac{d\boldsymbol{p_i}}{d\tau} &= -\frac{\partial H}{\partial \boldsymbol{\omega}_i} = -\frac{\partial E}{\partial \boldsymbol{\omega}_i}.\end{aligned}$$

These equations have some important properties:

1. If we follow the equations for some time, the Hamiltonian does not change ($\frac{dH}{d\tau} = 0$).
2. They are invertible: suppose we take a certain $(\boldsymbol{\omega}, \boldsymbol{p})$ combination as start position and we follow the equations for some time Δt and end up at $(\boldsymbol{\omega}', \boldsymbol{p}')$. If we now start from $(\boldsymbol{\omega}', -\boldsymbol{p}')$, we will end up in $(\boldsymbol{\omega}, -\boldsymbol{p})$.
3. A certain volume will stay the same under the transformation, because the divergence is zero.

We will generate new states in the chain by first generating a momentum vector $\boldsymbol{p}$ according to $\mathrm{p}(\boldsymbol{p}) \propto e^{-K(\boldsymbol{p})}$; the components of $\boldsymbol{p}$ simply have a standard normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\mathcal{I}})$. Using these $\boldsymbol{p}$-values together with our network parametrization $\boldsymbol{\omega}$, we will follow the Hamiltonian equations for some time Δt. The place where we end up will be our next state $(\boldsymbol{\omega}', \boldsymbol{p}')$ in the Markov chain, from which we will retain only $\boldsymbol{\omega}'$. We denote this procedure to generate $\boldsymbol{\omega}'$ from $\boldsymbol{\omega}$ with a (deterministic) distribution as $T_{\Delta t}(\,\boldsymbol{\omega}' \,|\, \boldsymbol{\omega}\,)$.

We can prove that this Markov chain has the desired equilibrium distribution $\mathrm{p}(\,\boldsymbol{\omega} \,|\, \mathbf{D}\,)$ by proving Equation 6.6:

$$\begin{aligned}T_{\Delta t}(\,\boldsymbol{\omega}' \,|\, \boldsymbol{\omega}\,)\mathrm{p}(\,\boldsymbol{\omega} \,|\, \mathbf{D}\,) &= \int_{\boldsymbol{p}'} T_{\Delta t}(\,\boldsymbol{\omega}', \boldsymbol{p}' \,|\, \boldsymbol{\omega}\,)\, d\boldsymbol{p}' e^{-E(\boldsymbol{\omega})}\\ &= \int_{\boldsymbol{p}'}\int_{\boldsymbol{p}} T_{\Delta t}(\,\boldsymbol{\omega}', \boldsymbol{p}' \,|\, \boldsymbol{\omega}, \boldsymbol{p}\,)\, e^{-K(\boldsymbol{p})} e^{-E(\boldsymbol{\omega})} d\boldsymbol{p}\, d\boldsymbol{p}'\\ &= \int_{\boldsymbol{p}'}\int_{\boldsymbol{p}} T_{\Delta t}(\,\boldsymbol{\omega}, -\boldsymbol{p} \,|\, \boldsymbol{\omega}', -\boldsymbol{p}'\,)\, e^{-H(\boldsymbol{\omega}, \boldsymbol{p})} d\boldsymbol{p}\, d\boldsymbol{p}'\\ &= \int_{\boldsymbol{p}'}\int_{\boldsymbol{p}} T_{\Delta t}(\,\boldsymbol{\omega}, -\boldsymbol{p} \,|\, \boldsymbol{\omega}', -\boldsymbol{p}'\,)\, e^{-H(\boldsymbol{\omega}', -\boldsymbol{p}')} d\boldsymbol{p}\, d\boldsymbol{p}'\\ &= T_{\Delta t}(\,\boldsymbol{\omega} \,|\, \boldsymbol{\omega}'\,)\, \mathrm{p}(\,\boldsymbol{\omega}' \,|\, \mathbf{D}\,).\end{aligned}$$

We used the reversibility of the Hamiltonian dynamics, and the fact that the Hamiltonian does not change while following these dynamics and thus has the same value in $(\boldsymbol{\omega}, \boldsymbol{p})$ as in $(\boldsymbol{\omega}', \boldsymbol{p}')$.

If we follow the dynamics for a time equal to zero, we can interpret $T_0(\,\boldsymbol{x},\boldsymbol{y}\,|\,\boldsymbol{\omega},\boldsymbol{p}\,)$ as a two-dimensional Dirac distribution, the probability that we are in $(\boldsymbol{\omega},\boldsymbol{p})$ if we follow the dynamics for a time equal to zero and started in $(\boldsymbol{\omega},\boldsymbol{p})$:

$$\int_{\boldsymbol{\Omega}}\int_{\mathbb{R}^n} f(\boldsymbol{x},\boldsymbol{y})\,T_0(\,\boldsymbol{x},\boldsymbol{y}\,|\,\boldsymbol{\omega},\boldsymbol{p}\,)\,d\boldsymbol{x}\,d\boldsymbol{y} = f(\boldsymbol{\omega},\boldsymbol{p}).$$

Because the volume stays the same under the dynamics, $T_{\Delta t}(\,\boldsymbol{x},\boldsymbol{y}\,|\,\boldsymbol{\omega},\boldsymbol{p}\,)$ stays a Dirac distribution, but now around the endpoints $(\boldsymbol{\omega}',\boldsymbol{p}')$ of our dynamics:

$$\int_{\boldsymbol{\Omega}}\int_{\mathbb{R}^n} f(\boldsymbol{x},\boldsymbol{y})\,T_{\Delta t}(\,\boldsymbol{x},\boldsymbol{y}\,|\,\boldsymbol{\omega},\boldsymbol{p}\,)\,d\boldsymbol{x}\,d\boldsymbol{y} = f(\boldsymbol{\omega}',\boldsymbol{p}').$$

To apply this method, we have to follow the dynamics for a fixed period of time. In general, this is only possible using a numerical approximation, which will introduce errors. As such, the Hamiltonian will change slightly during the process. We can deal with this problem by combining the hybrid Monte Carlo method with the Metropolis algorithm: we will generate new *candidate* states for the Metropolis algorithm by following the Hamiltonian dynamics and accept each new state from the numerical approximation of the Hamiltonian dynamics as a valid state using the Metropolis method. If $H(\boldsymbol{\omega},\boldsymbol{p})$ decreases or stays the same, the new state will always be accepted. If it increases, we can still accept the new state with a probability equal to the ratio of the Boltzmann densities.

The nice thing is that we should not care about making approximation errors, the Metropolis part will take care of this. The only thing we have to take care of, is that we follow the equations in symmetric manner: we have to be able to reverse the *numerical* trajectory. If we make some numerical errors in one direction, we have to make *the same* numerical errors if we go in the other direction. To achieve this, we will not make use of some widely applied Runge-Kutta method, but discretize the equations by using *leapfrog* steps:

$$\begin{aligned}
\hat{\boldsymbol{p}}_i(\tau+\frac{\varepsilon}{2}) &= \hat{\boldsymbol{p}}_i(\tau) - \frac{\varepsilon}{2}\frac{\partial E}{\partial \boldsymbol{\omega}_i}(\hat{\boldsymbol{\omega}}(\tau)) \\
\hat{\boldsymbol{\omega}}_i(\tau+\varepsilon) &= \hat{\boldsymbol{\omega}}_i(\tau) + \varepsilon\frac{\hat{\boldsymbol{p}}_i(\tau+\frac{\varepsilon}{2})}{m_i} \\
\hat{\boldsymbol{p}}_i(\tau+\varepsilon) &= \hat{\boldsymbol{p}}_i(\tau+\frac{\varepsilon}{2}) - \frac{\varepsilon}{2}\frac{\partial E}{\partial \boldsymbol{\omega}_i}(\hat{\boldsymbol{\omega}}(\tau+\varepsilon)).
\end{aligned}$$

Such leapfrog steps exist of a half step for $\boldsymbol{p}_i$, a full step for $\boldsymbol{\omega}_i$ and again a half step for $\boldsymbol{p}_i$. To follow these equations during a period Δt, we try to choose the step size ε as large as possible while keeping the approximation error reasonable and thus keeping the Metropolis acceptance rate reasonable. The above equations are then applied $L = \Delta t/\varepsilon$ times. From these equations, it is apparent that we can combine the half steps for $\boldsymbol{p}_i$ except for the first and the last step.

Additionally, we try to set Δt as large as possible such that $(\boldsymbol{\omega},\boldsymbol{p})$ and $(\boldsymbol{\omega}',\boldsymbol{p}')$ are far from each other and we walk through the parameter space with a high speed.

To estimate how large we can set ε and Δt, we run the Markov chain while optimizing these parameters to get a suitable acceptance rate. Once they are fixed, we can start using our Markov chain to generate samples from the posterior.

6.8 Prior for parameters

Before we can use all our machinery to sample from the posterior distribution, we should define the prior distribution. This distribution contains our belief in the density of the weight vector $\boldsymbol{\omega}$, before we observe any data. It is hard to give a clear interpretation to these weights in general, which has hindered so far the specification of the prior. We will discuss a range of priors and how to specify them.

6.8.1 Complexity-based prior

The first and most basic prior has the purpose to restrict the network parameters from becoming too large, for reasons explained in Section 6.3.1. We choose for a multivariate Gaussian distribution $\mathrm{p}(\,\boldsymbol{\omega}\,|\,\xi\,) = \mathcal{N}(\mathbf{0}, {\sigma_p}^2\boldsymbol{\mathcal{I}})$ for its nice properties and interpretation as a *weight decay* regularization term, as explained in Section 6.3.3. The standard deviation σ_p controls the freedom of the network weights and hereby controls the complexity of the network functions that can be reached. As an example, Figure 6.10 displays network functions drawn from Gaussian distributions with ${\sigma_p}^2$ ranging from 0.1 to 100. The network structure is the same for each prior and contains one hidden layer with 10 nodes. The hidden neurons use the hyperbolic tangent transfer function, while the output node uses the linear transfer function.

For networks drawn from a prior with increasing variance, the complexity increases accordingly from the *zero function* to a step function with a step for each hidden neuron.

Most of the time it is not clear how to set σ_p. A solution is to move everything one level up by defining a *hyper-distribution* for ${\sigma_p}^2$. A common choice is a Gamma prior for $u = {\sigma_p}^{-2}$ with mean m and shape parameter s

$$\mathrm{p}(\,u\,|\,s, m\,) = \frac{(s/(2m))^{s/2}}{\Gamma(s/2)} u^{s/2-1} e^{-us/(2m)}.$$

A small value for s will result in a vague prior for ${\sigma_p}^{-2}$, which is often desirable.

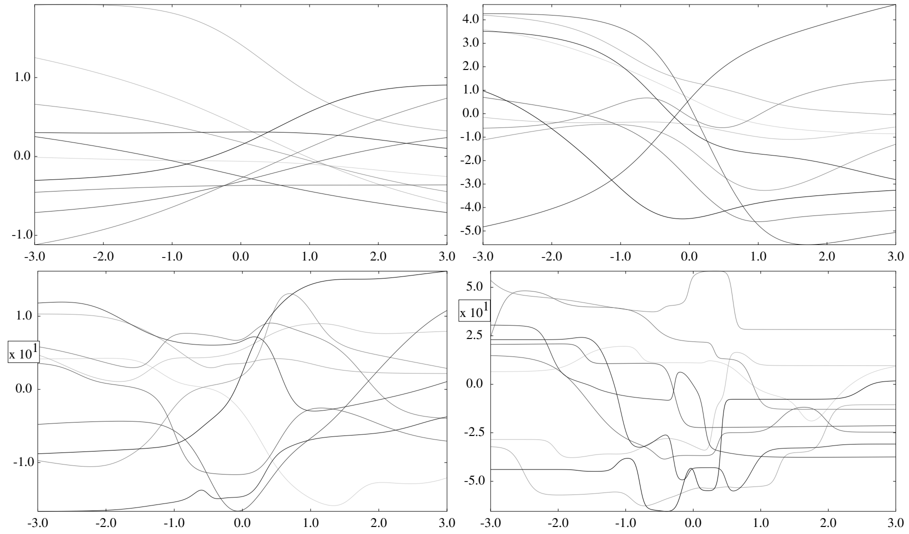


Figure 6.10: Network functions drawn from prior distributions with variance ${\sigma_p}^2$ ranging from 0.1 (left upper corner) to 100 (right bottom). The structure is four times the same, one hidden layer with 10 nodes.

The marginal of $\boldsymbol{\omega}$ can be computed exactly:

$$\begin{aligned}
\mathrm{p}(\,\boldsymbol{\omega}\,|\,s,m\,) &= \int_0^\infty \mathrm{p}(\,\boldsymbol{\omega}\,|\,{\sigma_p}^{-2}=u\,)\,\mathrm{p}(\,u\,|\,s,m\,)\,du \\
&= \int_0^\infty \frac{u^{n/2}}{\sqrt{2\pi}^n} e^{-u\sum_i \boldsymbol{\omega}_i^2/2}\,\frac{(s/2m)^{s/2}}{\Gamma(s/2)} u^{s/2-1} e^{-us/2m} du \\
&= \frac{(s/2m)^{s/2}}{\sqrt{2\pi}^n\Gamma(s/2)} \int_0^\infty u^{n/2+s/2-1} e^{-u(\sum_i \boldsymbol{\omega}_i^2/2+s/2m)} du \\
&= \frac{(s/2m)^{s/2}\,\Gamma(n/2+s/2)}{\sqrt{2\pi}^n\Gamma(s/2)(\sum_i \boldsymbol{\omega}_i^2/2+s/2m)^{n/2+s/2}} \\
&= \frac{\Gamma(n/2+s/2)}{\sqrt{\pi s/m}^n\,\Gamma(s/2)} \left(\frac{\sum_i \boldsymbol{\omega}_i^2}{s/m}+1\right)^{-(n/2+s/2)} ,
\end{aligned}$$

which is a multivariate *t*-distribution and can equally well be used in the hybrid Markov Chain Monte Carlo method.

Other approaches use an improper prior for ${\sigma_p}^{-2}$ and approximates $\mathrm{p}(\,{\sigma_p}^{-2}\,|\,\mathbf{D}\,)$

with a Dirac function around its peak ${\sigma_{p_*}}^{-2}$:

$$\begin{aligned}
\mathrm{E}[\,f(\cdot)\,|\,\mathbf{D}\,] &= \int_{\boldsymbol{\Omega}} f(\boldsymbol{\omega})\,\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)\,d\boldsymbol{\omega} \\
&= \int_{\boldsymbol{\Omega}}\int_{{\sigma_p}^{-2}} f(\boldsymbol{\omega})\,\mathrm{p}(\,\boldsymbol{\omega}\,|\,\mathbf{D}, {\sigma_p}^{-2}\,)\,\mathrm{p}(\,{\sigma_p}^{-2}\,|\,\mathbf{D}\,)\,d{\sigma_p}^{-2} d\boldsymbol{\omega} \\
&\approx \int_{\boldsymbol{\Omega}} f(\boldsymbol{\omega})\,\frac{\mathcal{L}(\,\boldsymbol{\omega}\,|\,\mathbf{D}\,)\mathrm{p}(\,\boldsymbol{\omega}\,|\,{\sigma_{p_*}}^{-2}\,)}{\mathrm{p}(\,\mathbf{D}\,|\,{\sigma_{p_*}}^{-2}\,)}\,d\boldsymbol{\omega}.
\end{aligned}$$

The *evidence framework* [60] describes a subtle approach to how to derive this value in terms of the effective number of parameters. A cruder approximation of ${\sigma_{p_*}}^{-2}$ can be found by maximizing $\mathrm{p}(\,\boldsymbol{\omega}, {\sigma_p}^{-2}\,|\,\mathbf{D}\,)$ both in $\boldsymbol{\omega}$ and σ_p, and retaining only the value for ${\sigma_p}^{-2}$:

$$\begin{aligned}
\sigma_{p_*} &= \mathrm{argmax}_{\sigma_p}(\mathrm{p}(\,{\sigma_p}^{-2}\,|\,\mathbf{D}\,)) \\
&= \mathrm{argmax}_{\sigma_p}(\int_{\boldsymbol{\Omega}} \mathrm{p}(\,\boldsymbol{\omega}, {\sigma_p}^{-2}\,|\,\mathbf{D}\,)) \\
&\approx \mathrm{argmax}_{\boldsymbol{\omega},\sigma_p}(\mathrm{p}(\,\boldsymbol{\omega}, {\sigma_p}^{-2}\,|\,\mathbf{D}\,))\big|_{\sigma_p}.
\end{aligned}$$

6.8.2 Informative prior

If we have more prior information than only the complexity of our network, we would like to use a distribution that is capable of expressing this information. Suppose that we somehow know in advance where the network parameters more or less have to be, we would at least need a distribution where we can specify the mean $\boldsymbol{\mu}$. If we have no clue about the correlations between the weights, we can simply use an independent Gaussian distribution $\mathrm{p}(\boldsymbol{\omega}) \sim \mathcal{N}(\boldsymbol{\mu}, \sigma\boldsymbol{\mathcal{I}})$ for each weight, possibly with the same variance σ^2 for each weight.

If more flexibility is necessary, we can specify correlations between the individual weights using a multivariate Gaussian distribution with a certain covariance matrix:

$$\mathrm{p}(\,\boldsymbol{\omega}\,|\,\boldsymbol{\mu}, \boldsymbol{\Sigma}\,) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}}\,e^{-\frac{1}{2}(\boldsymbol{\omega}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{\omega}-\boldsymbol{\mu})}.$$

As explained in Section 5.3.2, these distributions can be represented with a continuous Bayesian network with linear-Gaussian variables. The independent model corresponds to a network without edges, while each multivariate Gaussian can always be represented by a fully connected Bayesian network structure $\mathcal{S}_{\mathrm{bn}}^{\mathrm{C}}$. A sparse network structure corresponds to a model where interaction between only some of the weights is present. If these linear dependencies are not adequate, one can always use a nonlinear-Gaussian Bayesian network (see Section 5.3.3) to capture the prior.

All the previous distributions are unimodal. If this is a restriction to specify the prior, a mixture of Gaussians could be used, which corresponds to a Bayesian network with hidden variables. These hidden nodes can be used to divide and separately control different weights in a neural network. Figure 6.11 presents such a Bayesian network structure where each node *Hidden** of the Bayesian network controls the weights coming in and out a hidden neuron of the neural network.

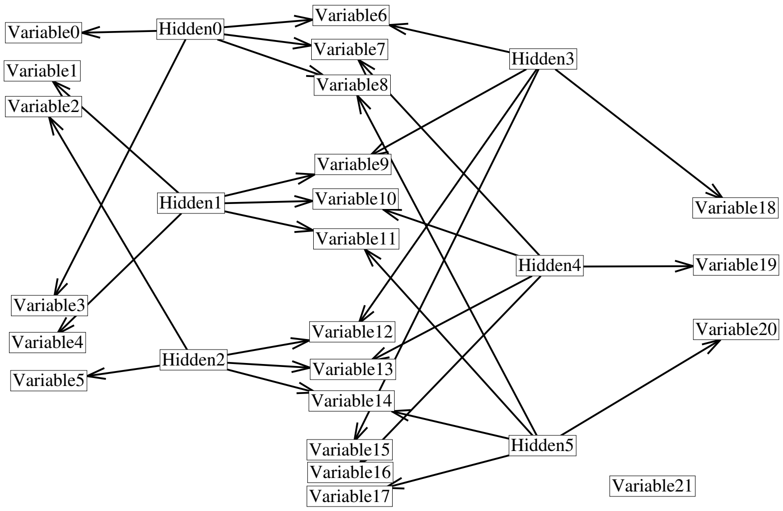


Figure 6.11: The fixed structure of the nonlinear-Gaussian Bayesian network that can be used to specify a prior for the neural network shown in Figure 6.12. Each hidden node controls one hidden neuron (the weights coming in and out of that neuron) of the multilayer perceptron.

The neural network structure has two inputs, two layers of hidden neurons with three neurons in each layer and one output, as shown in Figure 6.12. The Bayesian network node *Hidden0* corresponds to the top neuron of the first hidden layer. *Variable0* and *Variable1* correspond to the weights that connect the inputs x_1 and x_2, while *Variable2* corresponds to the bias that connects the input 1 to this neuron. *Variable6*, *Variable7*, and *Variable8* corrspond to the weights that connect this top neuron form the first layer to the neurons of the second layer.

By modifying the value of a hidden node of this Bayesian network, one can increase or decrease all the corresponding weights of the neural network, hereby modifying only the transfer function of that neuron. By dividing the weights in groups of separately controlled variables, one can modify the network function only locally. This can be important if a prior can be specified which has the same characteristics as the data set, but is still different, as explained in Section 4.5. Optimally, the prior successfully uses the neurons to describe the required characteristics of the problem. When data is observed, much of the prior can be kept, only the weights of some neurons will have to increase or

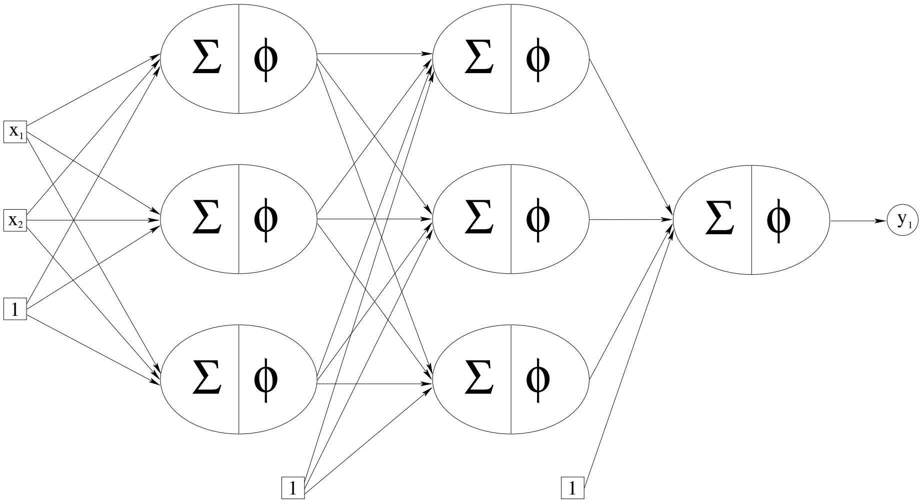


Figure 6.12: A neural network structure with two layers of hidden neurons. A possible Bayesian network structure with hidden nodes that can be used to specify a prior distribution is shown in Figure 6.11.

decrease simultaneously to control how smooth the neuron reacts on the inputs. How the different inputs of each neuron are combined in the neuron basically stays the same. The usage of the hidden nodes allows.

To use one of these priors in the hybrid Monte Carlo Markov chain procedure, we have to be able to compute the derivative of the prior. Although with increasing computational complexity, we can compute the gradient for each distribution that has been mentioned here. But before we can compute any of these gradients, we have to specify them.

6.9 Estimation of prior

As has been mentioned several times, we should not hope to specify an informative prior distribution directly, because of the difficult interpretation of the neural network weights. To incorporate the prior information anyway, we will apply the techniques proposed in Chapter 4. We assume that we can specify our prior knowledge by defining a Bayesian network structure together with an informative distribution (the *donor* model). By generating virtual data sets $\mathbf{D}_k$ and drawing parametrizations from the resulting neural network posterior distribution, we can generate weight vectors according to the informative prior distribution $\mathrm{p}(\boldsymbol{\omega} \,|\, \xi)$. One possibility to perform Bayesian inference using the posterior $\mathrm{p}(\boldsymbol{\omega} \,|\, \mathbf{D}, \xi)$ is to directly apply the rejection sampling method discussed in Section 6.7.1, but, although this informative prior is less broad than a complexity-based prior, this method will in most cases still not work efficiently.

Another approach is to generate a set of multilayer perceptron parametrizations $\{\boldsymbol{\omega}_i\}_{i=1}^d$ from $\mathrm{p}(\,\boldsymbol{\omega}\,|\,\xi\,)$, and estimate this distribution using the generated samples with one of the distributions mentioned in the previous section. Although it is straightforward to estimate one of these models on a statistical data set, we have to take care of some particular properties of neural network weight space: *symmetries.* There may exist different network parametrizations $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ that result in exactly the same network functions $\mathrm{mlp}(\,\boldsymbol{x}\,|\,\boldsymbol{\omega}_1\,) = \mathrm{mlp}(\,\boldsymbol{x}\,|\,\boldsymbol{\omega}_2\,),\ \forall \boldsymbol{x}$. Those symmetries are no problem when optimizing the posterior distribution or sampling from it, but they do become a problem when we want to estimate a distribution in weight space: two network parametrizations might be close to each other, while some of their symmetric representatives might be far from each other. Because we do not know which symmetric representative has been chosen for each parametrization, we might end up with poor estimations.

6.9.1 Cause of symmetries

There are two main reasons for these symmetries, as illustrated schematically in Figure 6.13. At first, if we swap two nodes from a hidden layer, the parametrizations will be different but the network function stays the same. The second type of symmetries is caused by the symmetry in the transfer function. The hyperbolic tangent transfer function is odd, which means $\phi(-x) = -\phi(x)$. If we flip the sign of all the weights that come in and out of a neuron in a hidden layer, the resulting network function will also stay the same. The total number of symmetries is

$$\#\text{Symmetries} = \prod_k 2^{N_k} N_k!,$$

where k goes over the hidden layers and N_k is the number of neurons in layer k.

6.9.2 Symmetry elimination methods

If we try to estimate a distribution in weight space, we are in fact estimating the superposition of #Symmetries distributions. It should be clear that we do not want to estimate each distribution separately because they are the same up to some transformation. Instead, we will try to choose for each network parametrization $\boldsymbol{\omega}$ that was generated from our informative prior, a suitable transformation $T(\boldsymbol{\omega})$ such that $\{T(\boldsymbol{\omega}_i)\}_{i=1}^d$ can directly be used to estimate the distribution in weight space.

Canonical

A first method to define such a transformation, is the *canonical* transformation. This transformation will choose for each hidden node thát sign flip that makes

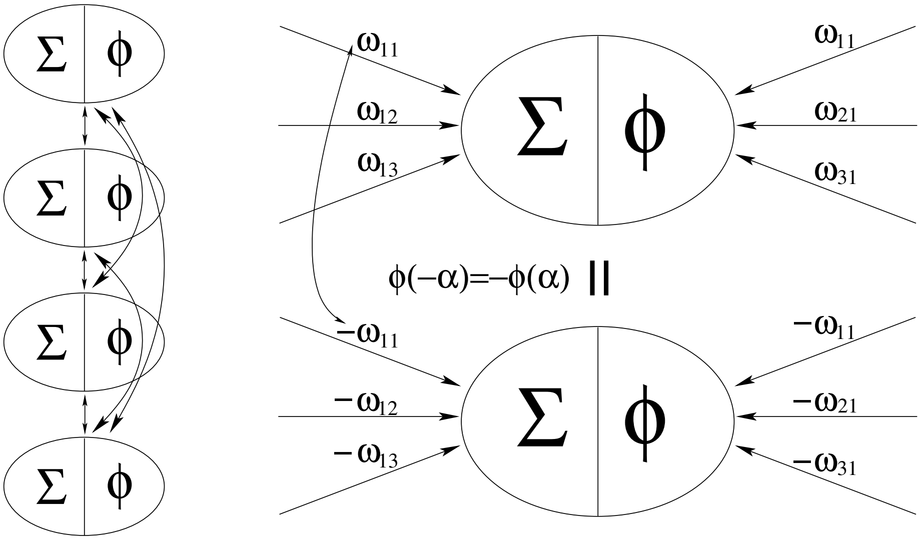


Figure 6.13: The two causes for symmetries in neural network weight space: node switch and transfer function symmetry.

the bias positive. Once all the biases are positive, the canonical transformation will order the biases from large to small to deal with the node swapping symmetries.

This transformation has the benefit that it maps each possible representation to a unique one. Unfortunately, this transformation has some discontinuities; suppose $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ are almost the same except that $\boldsymbol{\omega}_1$ has some bias slightly positive, while $\boldsymbol{\omega}_2$ has that same bias slightly negative. The canonical transformation will flip the signs of all the weights connected to the same node as that bias, which results in two very different network parametrizations $T_c(\boldsymbol{\omega}_1)$ and $T_c(\boldsymbol{\omega}_2)$. This will result in suboptimal density estimation.

Exact

To find more suitable transformations, we will first state more precisely what we are looking for. Suppose we want to estimate our distribution with a parametric density model $\mathrm{p}(\,\boldsymbol{\omega}\,|\,\boldsymbol{\nu}\,)$, where $\boldsymbol{\nu}$ denote the parameters of that model. This model can be one of the models indicated in Section 6.8.2 or some other multivariate model where the likelihood function $\mathcal{L}(\,\boldsymbol{\nu}\,|\,\{T_i(\boldsymbol{\omega}_i)\}_{i=1}^{d}\,)$ can be easily optimized with respect to the model parameters. We are now looking for those transformations $\{T_i(\cdot)\}_{i=1}^{d}$ that result in a set of transformed network parametrizations $\{T_i(\boldsymbol{\omega}_i)\}_{i=1}^d$ that can be modelled well by our density:

$$\begin{aligned}\{T_i(\cdot)\}_{i=1}^d &= \mathrm{argmax}_{T_i}(\max_{\boldsymbol{\nu}} \mathrm{p}(\,\{T_i(\boldsymbol{\omega}_i)\}_{i=1}^d \,|\, \boldsymbol{\nu}\,)) \\ &= \mathrm{argmax}_{T_i}(\max_{\boldsymbol{\nu}} \mathcal{L}(\,\boldsymbol{\nu} \,|\, \{T_i(\boldsymbol{\omega}_i)\}_{i=1}^d\,)).\end{aligned}$$

Computing this in a straightforward manner is infeasible, as this would require $\#\mathrm{Symmetries}^d$ optimizations of the likelihood function. The first problem is the factor d.

Instead of blindly trying every possible combination of possible transformations for the network parametrizations, we will try a more intelligent approach. The idea is that we pick a certain parametrization $\boldsymbol{\nu}$ and try to find those transformations that result in network parametrizations with a high probability according to $\mathrm{p}(\,\cdot\,|\,\boldsymbol{\nu}\,)$. Since we picked one parametrization $\boldsymbol{\nu}$, we can find a good transformation for each network parametrizations *separately*, effectively exchanging the exponent d for a multiplication. Next, we reestimate $\boldsymbol{\nu}$ based on the suitable transformations we just found, and repeat the whole process. Although a bit technical, the previous idea can be seen as an application of the Expectation Maximization algorithm [10].

This EM algorithm is an optimization method for the likelihood when missing or hidden variables are present. If we denote the observed data by $\boldsymbol{\mathcal{X}}$, the missing or hidden data with $\boldsymbol{\mathcal{Y}}$ and $\boldsymbol{\theta}$ for the parameters of the joint distribution $\mathrm{p}(\,\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \,|\, \boldsymbol{\theta}\,)$, this algorithm searches for $\boldsymbol{\theta}^*$ such that

$$\begin{aligned}\boldsymbol{\theta}^* &= \mathrm{argmax}_{\boldsymbol{\theta}}(\mathrm{p}(\,\boldsymbol{\mathcal{X}} \,|\, \boldsymbol{\theta}\,)) \\ &= \mathrm{argmax}_{\boldsymbol{\theta}}(\mathcal{L}(\,\boldsymbol{\theta} \,|\, \boldsymbol{\mathcal{X}}\,)),\end{aligned}$$

by alternating two steps. In the first step — the Expectation step — we look for the expected complete-data log-likelihood function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^i)$, using the current parameters $\boldsymbol{\theta}^i$ and the observed data $\boldsymbol{\mathcal{X}}$

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^i) = \mathrm{E}[\,\log \mathcal{L}(\,\boldsymbol{\theta} \,|\, \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}\,) \,|\, \boldsymbol{\mathcal{X}}, \boldsymbol{\theta}^i\,].$$

In the next step — the Maximization step — we find our next parametrization, $\boldsymbol{\theta}^{i+1}$, by optimizing Q

$$\boldsymbol{\theta}^{i+1} = \mathrm{argmax}_{\boldsymbol{\theta}}(Q(\boldsymbol{\theta}, \boldsymbol{\theta}^i)).$$

It is guaranteed that the iterative application of these steps converge to at least a local maximum of the likelihood function $\mathcal{L}(\,\boldsymbol{\theta} \,|\, \boldsymbol{\mathcal{X}}\,)$.

It is tempting to choose our unknown transformations $\{T_i(\cdot)\}_{i=1}^d$ as the hidden data $\boldsymbol{\mathcal{Y}}$ and $\boldsymbol{\theta} = \boldsymbol{\nu}$, but this results in an impractical algorithm. Instead, we choose $\boldsymbol{\mathcal{Y}} = \boldsymbol{\nu}$ and $\boldsymbol{\theta} = \{T_i\}_{i=1}^d$. The observed data is $\boldsymbol{\mathcal{X}}$, which is in our case the set of network parametrizations $\{\boldsymbol{\omega}_i\}_{i=1}^d$. We can approximate Q as follows:

$$\begin{aligned}
Q(\boldsymbol{\theta}, \boldsymbol{\theta}^i) &= \mathrm{E}[\log \mathrm{p}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \,|\, \boldsymbol{\theta}) \,|\, \boldsymbol{\mathcal{X}}, \boldsymbol{\theta}^i] \\
&= \int \log \mathrm{p}(\{\boldsymbol{\omega}_k\}, \boldsymbol{\nu} \,|\, \{T_k\})\, \mathrm{p}(\boldsymbol{\nu} \,|\, \{\boldsymbol{\omega}_k\}, \{T_k^i\})\, d\boldsymbol{\nu} \\
&\approx \log \mathrm{p}(\{\boldsymbol{\omega}_k\}, \boldsymbol{\nu}^* \,|\, \{T_k\}) \\
&= \log \left(\frac{\mathrm{p}(\{\boldsymbol{\omega}_k\}, \{T_k\} \,|\, \boldsymbol{\nu}^*)\, \mathrm{p}(\boldsymbol{\nu}^* \,|\, \{T_k\})}{\mathrm{p}(\{T_k\})} \right) \\
&= \log(\mathrm{p}(\{\boldsymbol{\omega}_k\}, \{T_k\} \,|\, \boldsymbol{\nu}^*)) + \mathrm{C}^{\underline{\mathrm{st}}} \\
&= \log(\mathrm{p}(\{T_k(\boldsymbol{\omega}_k)\} \,|\, \boldsymbol{\nu}^*)) + \mathrm{C}^{\underline{\mathrm{st}}},
\end{aligned} \tag{6.7}$$

where we approximated $\mathrm{p}(\boldsymbol{\nu} \,|\, \{\boldsymbol{\omega}_k\}, \{T_k^i\})$ with a Dirac function

$$\begin{aligned}
\mathrm{p}(\boldsymbol{\nu} \,|\, \{\boldsymbol{\omega}_k\}, \{T_k^i\}) &\approx \delta_{\boldsymbol{\nu}^*}(\boldsymbol{\nu}) \\
\boldsymbol{\nu}^* &= \mathrm{argmax}_{\boldsymbol{\nu}}(\mathrm{p}(\boldsymbol{\nu} \,|\, \{\boldsymbol{\omega}_k\}, \{T_k^i\})).
\end{aligned}$$

In addition to this, we assume that we have no prior knowledge for the transformations. This results in a uniform distribution for $\mathrm{p}(\{T_k\})$. It is also sensible that the knowledge of these transformations alone without the actual network parametrizations will not influence the model parameters $\boldsymbol{\nu}$. This results in the constant factor of Equation 6.7.

Optimizing $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^i)$ is now a matter of finding those transformations that maximize the log-likelihood with model parameters $\boldsymbol{\nu}^*$. Finding these optimal transformations can be done one by one. In the brute force manner, this requires the evaluation of $\mathrm{p}(T(\boldsymbol{\omega}_i) \,|\, \boldsymbol{\nu}^*)$ for each of the #Symmetries possible transformations and for each weight vector $\boldsymbol{\omega}_i$. This results in a complexity of $d \cdot$ #Symmetries, which is already much better than our previous complexity of #Symmetries^d, but #Symmetries can still be too large in practice. Let us look into a heuristic to find these transformations.

Heuristic

At first, let us assume that we have only one hidden layer with hyperbolic tangent transfer functions and l nodes and want to use a distribution that treats the weights connected to different nodes independently. To make clear what we mean, we divide the network parameters $\boldsymbol{\omega}$ into $l+1$ groups $(\tilde{\boldsymbol{\omega}}_1, \ldots, \tilde{\boldsymbol{\omega}}_{l+1})$; the first l groups correspond to the weights belonging to the hidden nodes, while the last group contains the other parameters. As an example, $\tilde{\boldsymbol{\omega}}_1$ contains the weights of the connections from the inputs to the first hidden node, the weight from the bias to this node *and* the weights of the connections from this first hidden node to the output nodes. The last group $\tilde{\boldsymbol{\omega}}_{l+1}$ contains in our case only the weights of the connections from the bias to the output nodes.

Our assumption for the distribution of the weights becomes

$$\begin{aligned}
\mathrm{p}(\,\boldsymbol{\omega}\,|\,\boldsymbol{\nu}\,) &= \prod_{i=1}^{l} \mathrm{p}(\,\tilde{\boldsymbol{\omega}}_i\,|\,\boldsymbol{\nu}_i\,) \cdot \mathrm{p}(\,\tilde{\boldsymbol{\omega}}_{l+1}\,|\,\boldsymbol{\nu}_{l+1}\,) \\
\log \mathrm{p}(\,\boldsymbol{\omega}\,|\,\boldsymbol{\nu}\,) &= \sum_{i=1}^{l} \log \mathrm{p}(\,\tilde{\boldsymbol{\omega}}_i\,|\,\boldsymbol{\nu}_i\,) + \log \mathrm{p}(\,\tilde{\boldsymbol{\omega}}_{l+1}\,|\,\boldsymbol{\nu}_{l+1}\,).
\end{aligned}$$

The distribution $\mathrm{p}(\,\tilde{\boldsymbol{\omega}}_i\,|\,\boldsymbol{\nu}_j\,)$ describes the density for the parameters corresponding to the j^{th} hidden node evaluated with $\tilde{\boldsymbol{\omega}}_i$. If $i \neq j$, we are effectively switching nodes i and j.

As indicated in Section 6.9.1, a general transformation consists of a permutation π of the nodes and a sign flip for some nodes. This observation allows us to write out the maximization over all possible transformations as follows:

$$\max_T \log \mathrm{p}(\,T(\boldsymbol{\omega})\,|\,\boldsymbol{\nu}\,) = \max_\pi \sum_{i=1}^{l} \max_\pm \log \mathrm{p}(\,\pm\tilde{\boldsymbol{\omega}}_{\pi(i)}\,|\,\boldsymbol{\nu}_i\,) + \mathrm{C}^{\underline{\mathrm{st}}}. \tag{6.8}$$

The permutation π takes care of the node permutations, while $\max_\pm$ takes care of the node flips. The constant factor comes from the parameter set $\tilde{\boldsymbol{\omega}}_{l+1}$, which is unaffected by our transformation.

Rüger and Ossen [75] rewrite Equation 6.8 in terms of a minimization

$$\max_T \log \mathrm{p}(\,T(\boldsymbol{\omega})\,|\,\boldsymbol{\nu}\,) = \min_\pi \sum_{i=1}^{l} \min_\pm - \log \mathrm{p}(\,\pm\tilde{\boldsymbol{\omega}}_{\pi(i)}\,|\,\boldsymbol{\nu}_i\,) + \mathrm{C}^{\underline{\mathrm{st}}}, \tag{6.9}$$

and state that this is the general travelling salesman problem with cost K(i,j) to go from city i to j

$$K(i,j) = \min_\pm - \log \mathrm{p}(\,\pm\tilde{\boldsymbol{\omega}}_j\,|\,\boldsymbol{\nu}_i\,).$$

They continue by using the rich variety of approximate solutions for the travelling salesman problem to deal with the symmetry problem. Unfortunately, we fail to see the travelling salesman problem in Equation 6.9. In our opinion the minimization of Equation 6.9 is much harder because it goes over all possible permutations, while the travelling salesman goes only over those permutations that result in a tour.

One can translate the above problem to a travelling salesman problem with $2l$ cities though. We have two types of cities: $\{\mathrm{c}_1, \ldots, \mathrm{c}_l\}$ and $\{\mathrm{c}_1^*, \ldots, \mathrm{c}_l^*\}$. We define the cost between two cities as:

$$\begin{aligned}
K(\mathrm{c}_i^*, \mathrm{c}_j) &= \min_\pm - \log \mathrm{p}(\,\pm\tilde{\boldsymbol{\omega}}_j\,|\,\boldsymbol{\nu}_i\,) \\
K(\mathrm{c}_i, \mathrm{c}_j^*) &= 0 \\
K(\mathrm{c}_i, \mathrm{c}_j) &= \infty \\
K(\mathrm{c}_i^*, \mathrm{c}_j^*) &= \infty.
\end{aligned}$$

Because of the special definition of this cost function, we force that a tour alternates between cities of the two types. The links of the type $\mathrm{c}_i^* \to \mathrm{c}_j$ indicate the node permutation that has to be used. The links $\mathrm{c}_i \to \mathrm{c}_j^*$ all have the same cost and are thus unimportant. Unfortunately, a travelling salesman problem with $2l$ cities is much harder to solve than one with only l cities.

Therefore, to continue with our optimization of $Q(\boldsymbol{\theta}, \boldsymbol{\theta}_i)$, we designed a greedy search heuristic procedure to find an approximate solution of Equation 6.8. The basis of this heuristic is the observation that a hidden node becomes more important for the overall network function if its corresponding weights become larger. With this in mind, we first reparametrize $\boldsymbol{\nu}$ in such a way that the mean L_2 distances for the weights of each hidden node becomes ordered

$$\mathrm{E}[\,\|\tilde{\boldsymbol{\omega}}\| \,|\, \boldsymbol{\nu}_1\,] > \mathrm{E}[\,\|\tilde{\boldsymbol{\omega}}\| \,|\, \boldsymbol{\nu}_2\,] > \cdots > \mathrm{E}[\,\|\tilde{\boldsymbol{\omega}}\| \,|\, \boldsymbol{\nu}_l\,]. \tag{6.10}$$

Doing so, $\mathrm{p}(\,\cdot \,|\, \boldsymbol{\nu}_1\,)$ models the most important hidden node, $\mathrm{p}(\,\cdot \,|\, \boldsymbol{\nu}_2\,)$ will model the second most important node and so on. To find thát transformation T for a network parametrization $\boldsymbol{\omega}$ that maximizes the probability $\mathrm{p}(\,T(\boldsymbol{\omega}) \,|\, \boldsymbol{\nu}\,)$, we reorder the groups of $\boldsymbol{\omega}$ similarly from large to small

$$\|\tilde{\boldsymbol{\omega}}_1\| > \|\tilde{\boldsymbol{\omega}}_2\| > \cdots > \|\tilde{\boldsymbol{\omega}}_l\|. \tag{6.11}$$

Starting from this reordered distribution and parameter vector, we search for a suitable transformation. Because $\tilde{\boldsymbol{\omega}}_1$ looks like the most important part of our network, we make sure that this is placed as correctly as possible. We will try the first K node distributions $\mathrm{p}(\,\cdot \,|\, \boldsymbol{\nu}_i\,)$, $i \in \{1, \ldots, K\}$ to fit $\pm\tilde{\boldsymbol{\omega}}_1$ as well as possible. Next, we try the first K node distributions that are left to fit $\pm\tilde{\boldsymbol{\omega}}_2$ as well as possible, and so on. K is a parameter of the heuristic. If it is set to 1, only the L_2 norm of the node weights is taken into account together with possible sign flips to find a suitable transformation. The larger it gets, the more potential node distributions are tested, resulting in a better transformation at the expense of a higher computational complexity.

> Initialization: Reorder $\boldsymbol{\nu}$ and $\boldsymbol{\omega}$ such that Equation 6.10 and Equation 6.11 hold.
>
> For i over the hidden nodes, from 1 to l do
>
> 1. Find that node distribution j and sign ± from the first K still available distributions that has the maximal probability: $\max_{\pm} \mathrm{p}(\,\pm\tilde{\boldsymbol{\omega}}_i \,|\, \boldsymbol{\nu}_j\,)$ is maximal.
> 2. $\pi^{-1}(i) = j$

A comparison of the different methods

We will demonstrate the above techniques to estimate a distribution in neural network weight space on an artificial problem to indicate the difference between them. Based on a noisy sinus data set and a complexity-based prior, we can obtain the posterior distribution. The network structure we use, has two hidden layers with five hyperbolic tangent neurons in each layer and is shown in Figure 6.14. We will generate 100 random vectors $\{\boldsymbol{\omega}_i\}$ from this distribution using the hybrid Monte Carlo Markov chain method. To make sure that the network parametrizations are not coming from the same symmetric quadrant, we use one Markov chain run to generate only one $\boldsymbol{\omega}_i$, after which we restart the chain from a random vector. The noisy sinus data set and the generated regression networks are shown in Figure 6.15, together with the mean of those network functions.

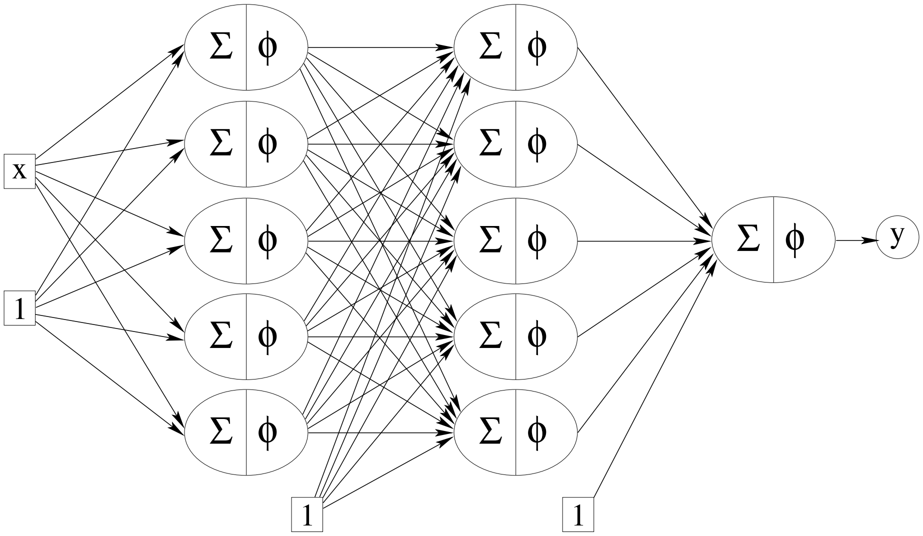


Figure 6.14: The structure of the multilayer perceptron we will use to model the sinus data set. This neural network consists of two hidden layers with five neurons in each layer, all using the hyperbolic tangent transfer function.

The original network parameter vectors are shown in Figure 6.16. Each column of this matrix is a network parametrization. We will try to estimate the distribution in weight space with a single Gaussian distribution using three different techniques: at first, we fit the distribution on the original weight vectors from Figure 6.16. From this figure, we can see that a single Gaussian distribution will not give a good performance because there are variables (rows) that have a bimodal behaviour and are either close to 2.6 (black) or -2.6 (white).

Next, we transformed the network parametrizations using the canonical transformation. The matrix on Figure 6.17 seems already easier to estimate, but improvements can still be made. This can be achieved by applying the greedy-based heuristic transformations. The resulting parameter vectors are shown on Figure 6.18.

From these three methods, the heuristic procedure seems the most suitable to fit a unimodal distribution on. This can also be seen from Figure 6.19,

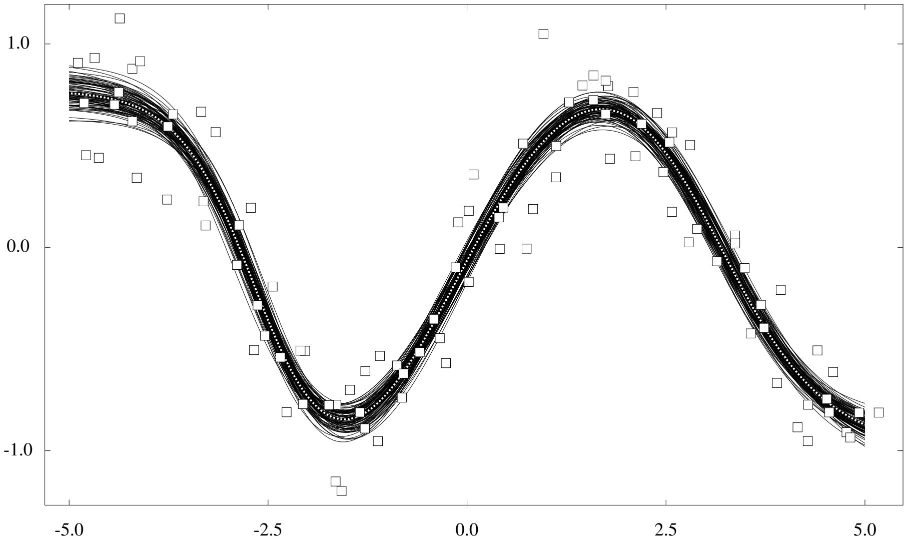


Figure 6.15: The noisy sinus data set and the functions realized by a few networks drawn from the posterior distribution, together with the mean of those network functions.

which shows a different scatter plot for each method between an input weight and a weight between the two hidden layers. We choose that weight that was connected to the most important node(s).

To evaluate the performance of these different methods, we used the estimated means of these three distributions as neural network parameter vectors and visualized their resulting input-output mappings on Figure 6.20. The mean of the Gaussian fitted on the original parameter set is almost the zero vector, which corresponds to the zero network mapping (the dotted line). This is as expected since it is equally likely for a weight to be positive or negative because of the sign flips. The canonical procedure is already much better (the dashed line) although the network function specified by the mean vector is not as it should be. The heuristic procedure (the full line) is clearly the best and follows the sinusoidal trend well.

6.10 To conclude

Neural networks have seen an explosion of interest over the last few decades resulting into many interesting results, both theoretical as practical. Dealing with neural networks in the Bayesian framework is more recent [63]. In this chapter, we described the most important properties of neural networks that we will use and indicated the similarities and differences between working with cost functions or performing Bayesian inference.

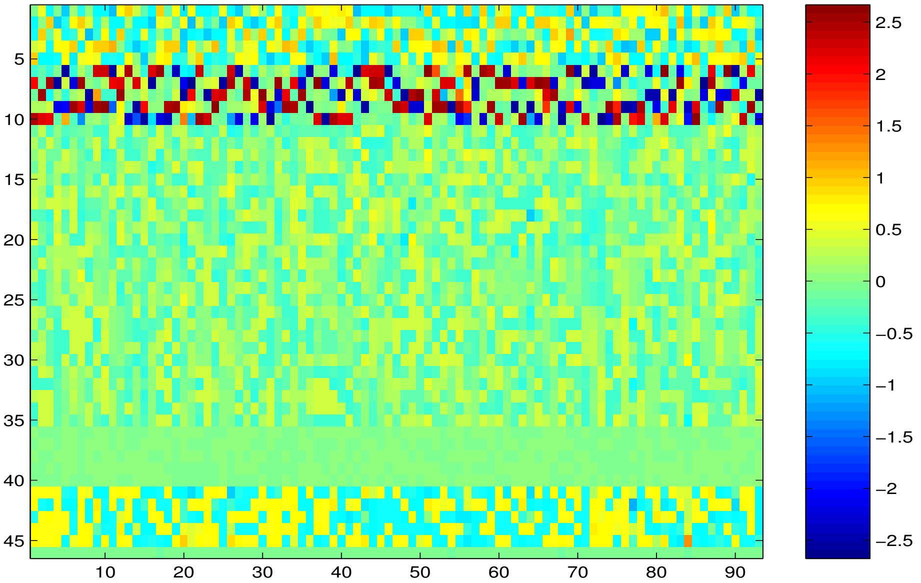


Figure 6.16: Each column from this matrix represents the weights from one neural network parametrization without any transformation. The neural network structure contained two hidden layers containing five hidden nodes each, as shown in Figure 6.14. The colour of each element from a column corresponds to a certain weight of the neural network model: the first five rows correspond to the weights connecting the input to the hidden layer, the following five rows connect the bias to these hidden nodes. Then come the connections from the first hidden layer to the next and the bias connections involved. At last, we have five weights connecting the nodes from the second hidden layer to the output and finally the bias weight.

To support the latter, we developed a method to estimate distributions in neural network weight space. We proposed a range of parametric distributions with increasing potential for successful estimation of a distribution and described the symmetry problem that arises when hidden nodes are present. To deal with them, we designed a method to divide a combinatorial problem into a set of smaller problems and developed a heuristic to deal with these smaller problems. This enabled us to do efficient density estimation in neural network weight space and will allow us to apply the transformation technique introduced in Chapter 4.

Estimation of these distributions in neural network weight space is necessary for dealing with the different types of information that are present in the ovarian tumour problem. We have now introduced the necessary ingredients to perform this task and will present the results in the following chapter.

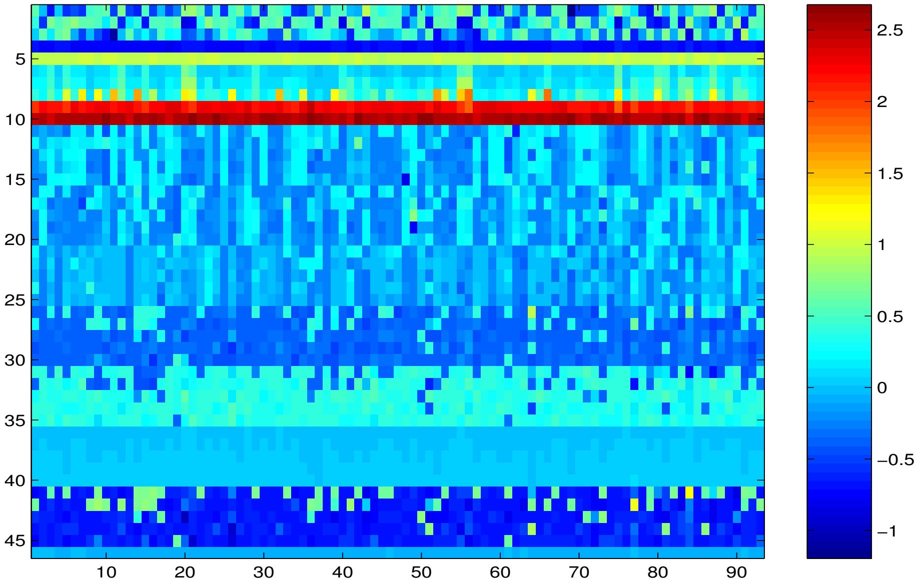


Figure 6.17: Each column from this matrix represents the weights from one neural network parametrization where we applied the canonical transformation.

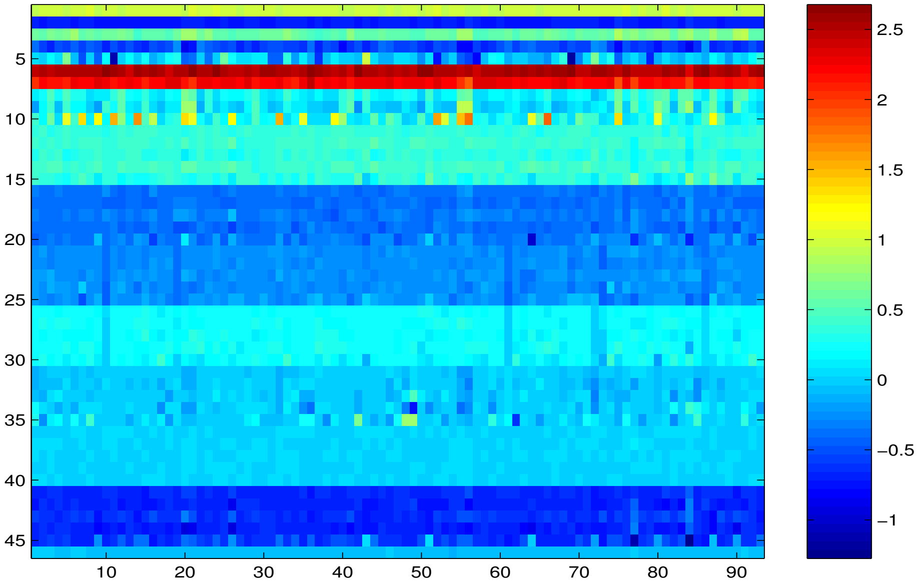


Figure 6.18: Each column from this matrix represents the weights from one neural network parametrization where we applied our heuristic procedure to find a suitable transformation.

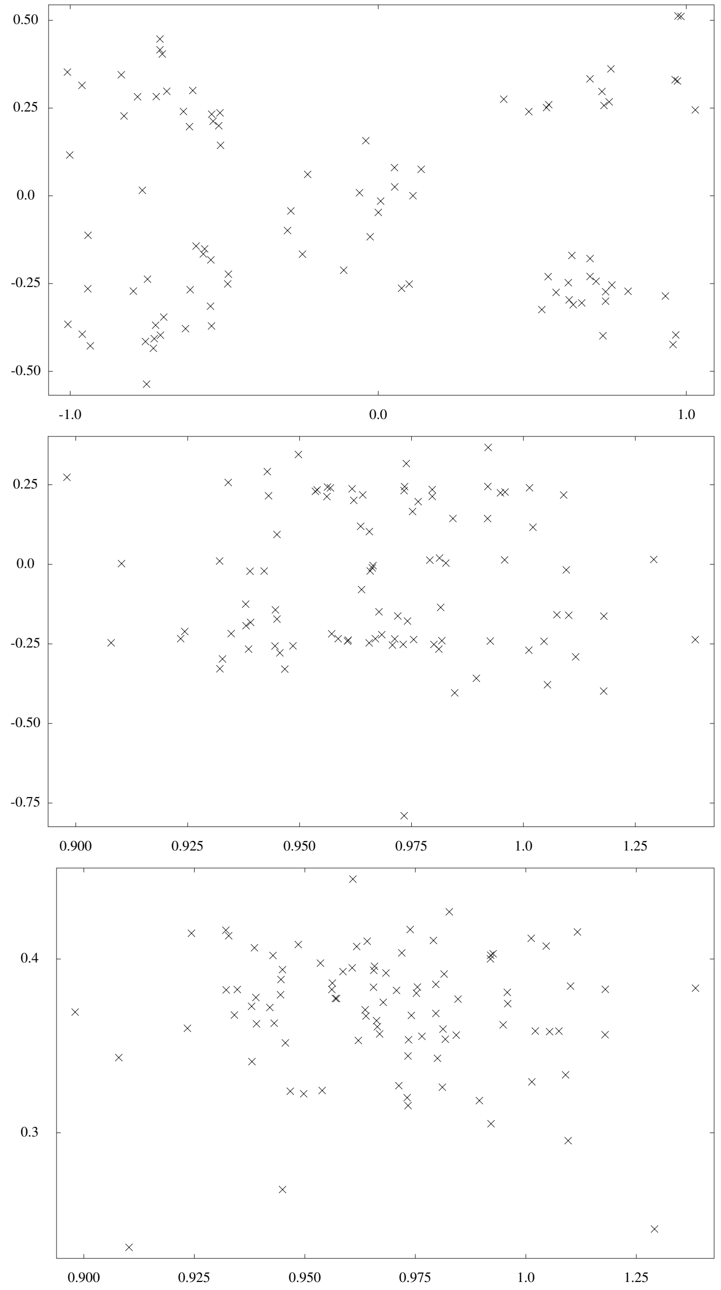


Figure 6.19: Three scatter plots indicating the difference between the symmetry elimination methods. On the x axis is the weight from the input to the most important node, on the y axis is the weight from the most important node in the first layer to the most important node in the second layer. Note the difference between the axis scales.

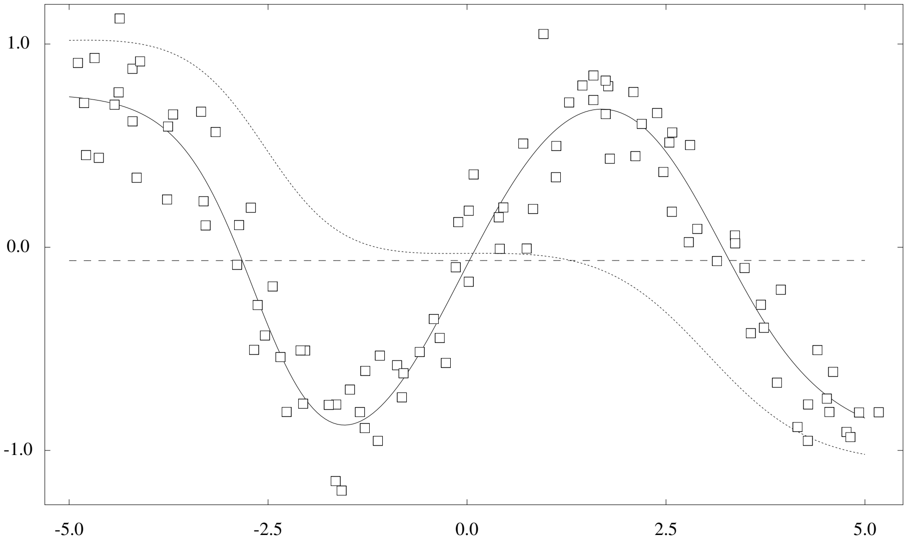
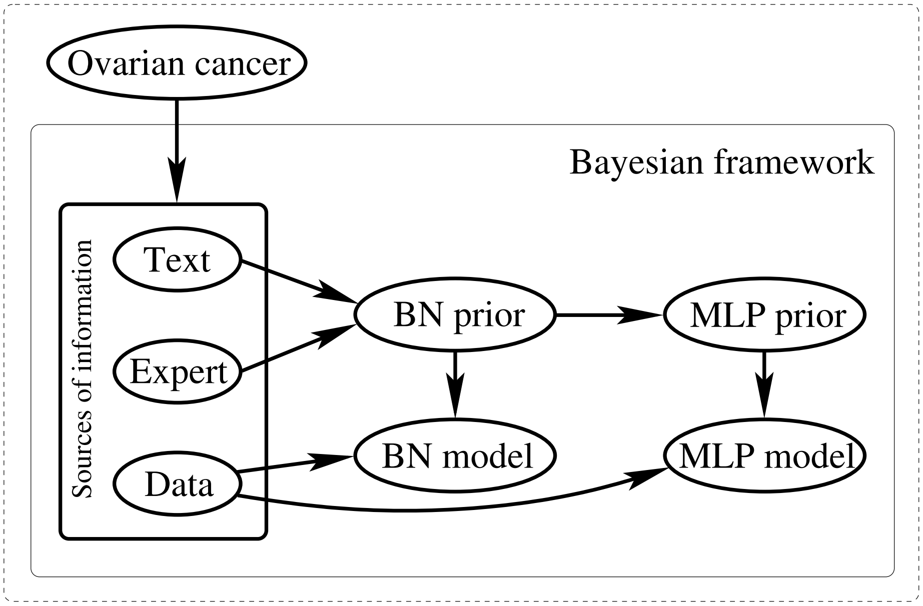


Figure 6.20: Based on the transformed network parametrizations, a Gaussian distribution is fitted. The three vectors of means resulting from the three different transformation methods are used as neural network parametrizations. The dashed line is the network function resulting from the mean estimated on the original parameter vectors. The dotted line comes from the canonical transformation, while the full line results from the heuristic procedure to find suitable transformations.

Chapter 7

Ovarian tumour classification



The previous three chapters were devoted to the description of the Bayesian network and multilayer perceptron model classes, their individual benefits and a possible strategy for how to combine these two different types of models.

Now that the techniques and methods are introduced, we will apply them to the ovarian tumour *problem that was briefly introduced in Chapter 2 and compare them to more traditional methods and previous work on this medical challenge. Recall that the goal was to predict pre-operatively if a certain ovarian tumour* *is benign or malignant, based on medical observations. In this chapter, we will develop several classification systems and compare their performance.*

7.1 Description

This work is done in the framework of the International Ovarian Tumour Analysis Consortium (IOTA), a study on ovarian tumours [83] initiated at the University Hospitals Leuven by Prof. Dr. Dirk Timmerman. When the IOTA project started, almost together with this research, there were only a few data samples available, which made it really necessary to try and include all possible other information that could be found. During the years, IOTA collected a lot of data, which makes it possible to learn good models based on numerical data alone. This simplifies somewhat the necessity to combine all possible kinds of information, but on the other hand allows us to test and validate the performance of our models better.

From this point of view, the ovarian tumour problem is not a *common* medical problem. Usually, only a small data set is available because of the effort (both time and cost) involved in collecting the data. In such a situation, combining all available kinds of information is a must.

7.2 Sources of information

Before we begin to develop our models, we introduce the data set and the additional information sources we have access to.

7.2.1 Clinical data

The original IOTA data set contains 68 parameters and 1 346 tumour records for 1 152 patients.[1] With the help of Prof. Timmerman and previous studies, we selected 35 relevant variables and 782 records. These variables are introduced and explained in detail in Appendix A.

From these 1 346 tumours we kept only the dominant tumour for patients with a tumour on both the left and right side. Records with missing values for any of these 35 variables are also removed from the data set, except for a missing *Pill Use* value, for which we used an imputation model to complete the variable.

Based on this rudimentary data set, we created two different data sets. The first derived data set is a *discrete* data set, with the purpose to be used with the Bayesian network models. The discretization of the continuous parameters is performed with the help of the discretization intervals that are indicated in Appendix A. These intervals were designed by our expert and are straightforward

---

[1] Although this is not the final IOTA data set, it underwent a quality control and inconsistent records were corrected by Andrea Valek.

to use: the limits of these intervals indicate what value should be assigned to the discretized variable.

The second derived data set is a *continuous* data set, to be used with the multilayer perceptron models. Because of the nature of these multilayer perceptrons, things are more complicated here. The reasoning behind the following steps is given in detail in Section 6.5.

At first, we observe that a multilayer perceptron is an input-output model. Accordingly, we choose the *Pathology* variable as the output variable, also called the response variable. This variable indicates if the tumour was found to be benign or malignant after surgery. It is a binary variable with possible values 0 or 1, which are transformed to the possible values -1 and 1 for reasons explained in Section 6.3.3 and Section 6.2.3. A 0 (-1) indicates a benign tumour (class $\mathcal{C}_{\mathrm{N}}$), while a 1 indicates a malignant tumour (class $\mathcal{C}_{\mathrm{P}}$). 70.5% of the records belong to class $\mathcal{C}_{\mathrm{N}}$.

Next, we take a look at the input variables, which are the other 34 parameters from the data set. The first type of variables that are present in the data set, are *binary* variables, discrete variables which can have only two possible values. These are easy to deal with, as we only transform their possible values 0 and 1 to -1 and 1 for symmetry reasons, just as we did with the *Pathology* variable.

Further, we have *discrete* variables with more than two possible values. These can be divided again into two different groups. At first, we have the *ordered* variables, variables where a natural order between the values exist. The values for such a variable are transformed in such a way that they become equally spreaded between -1 and 1. To clarify what we mean, lets take the *Colour Score* variable. This variable has four possible values: 1, 2, 3 and 4, where the numerical value indicates the amount of blood flow through the tumour. We will transform these values to -1, -1/3, 1/3 and 1.

The other group of discrete variables are the *nominal* variables; their values do not have a natural ordering. This makes it hard to represent them with one variable in a logistic regression or multilayer perceptron setup. Therefore, we introduce a separate *design* variable for each different possible value.

The other category of variables, are the *continuous* variables. These variables take values in $\mathbb{R}$. Pre-processing these variables starts with possibly applying some transformations to construct a linear relation between the variable and the logarithm of the odds. We constructed logit plots for each continuous variable, as explained in Section 6.5.1 and shown in Appendix A. These logit plots are used to get insight into the relation between the variable at hand and the *Pathology* variable. If such a transformation was necessary, this was the logarithmic transformation.

After a possible transformation, we normalized all the continuous variables for unit variance in a linear way, by subtracting the sample mean and dividing by the standard deviation.

Finally, special care was taken for variables that contain a “not relevant” value: some variables are not applicable or not relevant in certain conditions. As such, the variable *CycleDay*, which represents the menstrual cycle, makes only sense for women who are pre-menopausal. In the same category, we have the *PMenoAge* and *PostMenoY* variables that only make sense for patients after the menopause. A second group deals with solid papillary projections. When these structures are present, the shape and possible flow is indicated using the variables *PapSmooth* and *PapFlow*. The last group of conditional variables deals with blood flow indices which are measured using colour Doppler imaging. This has the effect that the variables *RI*, *PI*, *PSV* and *TAMXV* are only valid when actual blood flow is present (*Colour Score* $\neq 1$).

We deal with such cases by setting the variable itself to zero and introducing an extra design variable which indicates "not relevant".

7.2.2 Expert information

An entirely different — but equally valuable — source of information came with the knowledge and experience of Prof. Timmerman, the gynaecological expert who initiated the IOTA project and collected a large part of the data records while working at the University Hospitals Leuven.

This information is less straightforward to use in quantitative models, and a lot of insight we gained in the field thanks to his experience, is hard to write down. As such, he helped selecting the relevant domain variables from the set of 68 parameters, described their meaning and relations, checked the models we learned and much more. We will describe two clearly defined pieces of information that were the most important ones for our experiments.

At first, Prof. Timmerman was able to express part of his knowledge as a Bayesian network. He selected 11 of the most important domain variables and constructed a Bayesian network structure expressing his conjecture regarding the conditional independencies that are valid among these variables. This structure can be seen in Figure 7.1.

After this structure was constructed, he could also specify a parametrization for this network, hereby effectively specifying a joint distribution for these 11 domain variables.

The reason that Prof. Timmerman restricted himself to 11 variables, was to still be able to specify a parametrization for the network. With more variables added to the network, it was much harder to specify a network structure and give a sensible parametrization. Some nodes had many parents, resulting in a huge number of parameters that have to be specified consistently. Even for this 11 node network, we constructed a question list to extract the different probabilities, containing 29 pages.

Additional pairwise relations between the set of 35 domain variables, could be specified using a matrix and the technique explained in Section 5.5.3. We asked to create a list of related variables for each variable. Once these lists were finished, an ordering between these connections was gradually constructed. This ordering was then again refined by attaching numbers to the importance of each individual relation. This results in a matrix $\boldsymbol{V}$ where element $\boldsymbol{V}_{ij}$ is an indication of the strength of the relation between the variables $\underline{x}_i$ and $\underline{x}_j$. We will use this matrix as the a priori probability that we see an edge from variable $\underline{x}_j$ to variable $\underline{x}_i$ in a Bayesian network. This matrix is displayed in Figure 7.2,

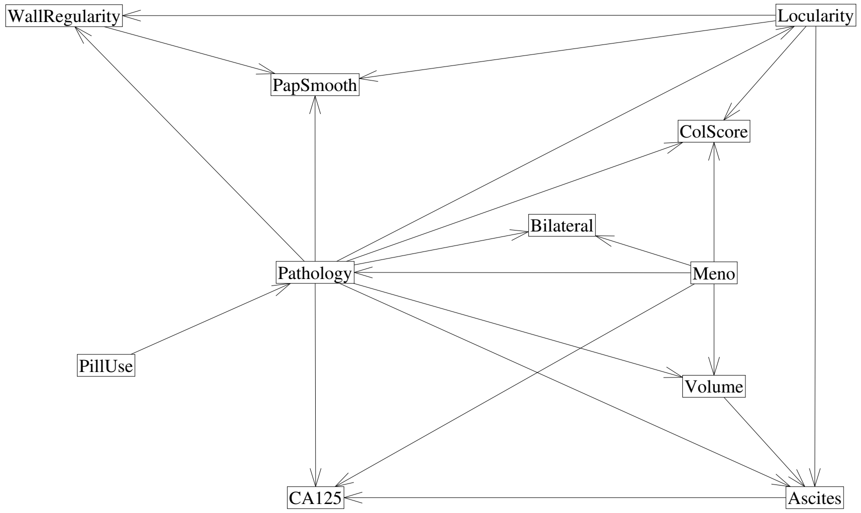


Figure 7.1: The network structure for the ovarian tumour problem containing 11 variables, constructed by Prof. Timmerman.

where the grey-scale and the thickness of an edge between $\underline{x}_i$ and $\underline{x}_j$ corresponds to the value $\boldsymbol{V}_{ij}$ from the matrix.

7.2.3 Textual information

A last and quite extensive source of information is contained in textual documents that report about the domain. We will use this literature information for an alternative method to discover pairwise relationships between domain variables, just as what we obtained from the expert in the form of the matrix $\boldsymbol{V}$ in the previous section.

We will describe two methods to convert the set of available documents to a pairwise relation matrix. The first method is based on variable annotations and derives the pairwise relations directly from these annotations. The second approach will use a large set of documents reporting about the domain and tries to find documents that describe two domain variables. The assumption here is that such documents most often talk about a relationship between these two variables. The more such documents that can be found, the stronger the relation between these two variables will be.

Let us start with the annotation-based method. We assume that each domain variable $\underline{x}$ is annotated with a textual description $\mathcal{D}(\underline{x})$. The textual descriptions $\mathcal{D}(\cdot)$ we used, consist of the concatenation of a set of keywords provided by the expert, the IOTA protocols, a description for each variable from a thesis [81] and definitions from the Merck Manual. In addition, we added

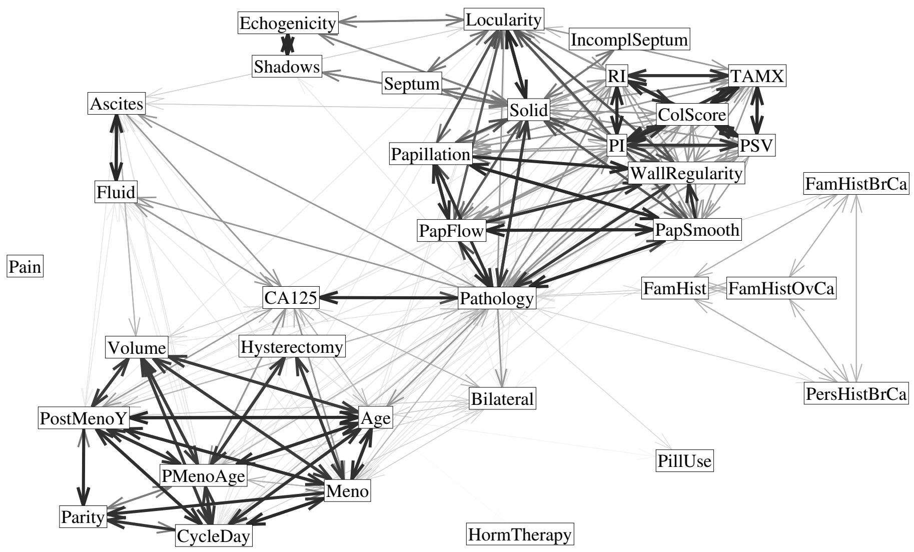


Figure 7.2: A representation of the matrix $\boldsymbol{V}$ where element $\boldsymbol{V}_{ij}$ is an indication of the strength of the relation between $\underline{x}_i$ and $\underline{x}_j$, constructed by Prof. Timmerman.

specific documents from the On-line Medical Dictionary, the CancerNet Dictionary and the MEDLINE collection of abstracts of the US National Library of Medicine to these annotations.

These per variable annotations are converted to a canonical representation using the Porter stemmer [26], a synonym list, and special phrase handling. Based on this canonical — but still textual format — a vector representation $\mathcal{T}(\mathcal{D}(\underline{x}))$ for each annotation is computed using the term-frequency inverse-document-frequency weighting scheme. The operation to convert a document into its vector representation is discussed in more detail in Section 5.5.3.

Based on the vector representations of the annotations of two variables $\underline{x}_j$ and $\underline{x}_k$, a similarity between them can be computed as the cosine of the angle between the two vectors:

$$\boldsymbol{V}_{\underline{x}_j\underline{x}_k} = \boldsymbol{V}_{\underline{x}_k\underline{x}_j} = \cos(\mathcal{T}(\mathcal{D}(\underline{x}_j)),\mathcal{T}(\mathcal{D}(\underline{x}_k))) = \frac{<\mathcal{T}(\mathcal{D}(\underline{x}_j)),\mathcal{T}(\mathcal{D}(\underline{x}_k))>}{\|\mathcal{T}(\mathcal{D}(\underline{x}_j))\|\|\mathcal{T}(\mathcal{D}(\underline{x}_k))\|}.$$

This similarity relation can directly be used to construct a pairwise relation matrix that can serve as a prior distribution for Bayesian network models. The general procedure to compute the matrix $\boldsymbol{V}$ is depicted in Figure 7.3.

The second approach is based on building a *binary presence* matrix $\boldsymbol{P}$ and deriving the pairwise relations from this matrix. To build this binary presence matrix, we need a corpus $\mathcal{C}$ of documents that talk about the domain. Each row in our binary presence matrix corresponds to one document, while each column corresponds to one specific domain variable. The value $\boldsymbol{P}_{ij}$ can be

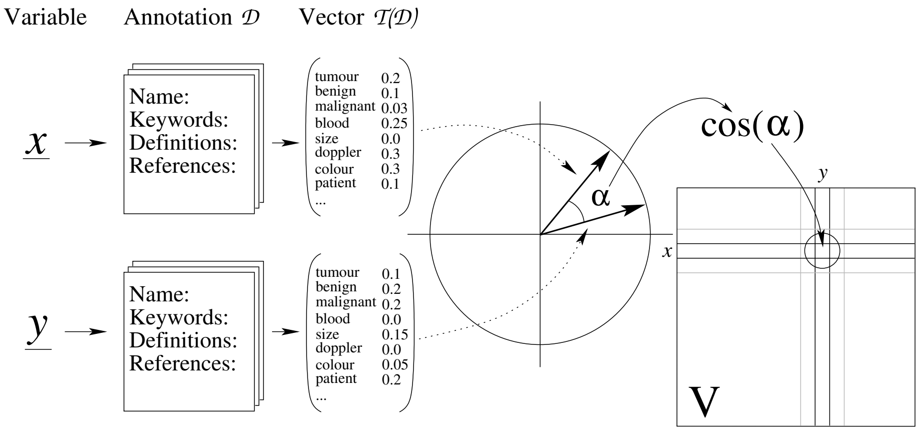


Figure 7.3: A schematic overview of the similarity-based approach to extract a relation matrix $\boldsymbol{V}$ from textual variable annotations. These variable annotations $\mathcal{D}$ are converted to their vector representations $\mathcal{T}(\mathcal{D})$ using the term-frequency inverse-document-frequency weighting scheme. The relation between two variables $\underline{x}$ and $\underline{y}$ is measured using the cosine between their respective vector representations.

either 0 or 1 and indicates if document $\mathcal{D}_i$ mentions domain variable $\underline{x}_j$. In our definition, a document $\mathcal{D}_i$ is said to mention a variable $\underline{x}_j$ if the similarity between the respective vector representations $\mathcal{T}(\mathcal{D}_i)$ and $\mathcal{T}(\mathcal{D}(\underline{x}_j))$ exceeds a certain threshold λ

$$\boldsymbol{P}_{ij} = \begin{cases} 1 & \text{if } \operatorname{sim}(\mathcal{T}(\mathcal{D}_i), \mathcal{T}(\underline{x}_j)) \geq \lambda \\ 0 & \text{else.} \end{cases}$$

This matrix can now be used to define pairwise relationships $\boldsymbol{V}_{\underline{x}_j\underline{x}_k}$ among the variables:

$$\begin{aligned} \boldsymbol{V}_{\underline{x}_j\underline{x}_k} = \boldsymbol{V}_{\underline{x}_k\underline{x}_j} &= \mathrm{P}(\,\underline{x}_j \in \mathcal{D} \text{ and } \underline{x}_k \in \mathcal{D} \,|\, \underline{x}_j \in \mathcal{D} \text{ or } \underline{x}_k \in \mathcal{D}, \mathcal{D} \in \mathcal{C}\,) \\ &= \frac{\sum_i \boldsymbol{P}_{ij}\boldsymbol{P}_{ik}}{\sum_i \boldsymbol{P}_{ij} + \sum_i \boldsymbol{P}_{ik} - \sum_i \boldsymbol{P}_{ij}\boldsymbol{P}_{ik}}. \end{aligned}$$

The corpus $\mathcal{C}$ comprises entries from the MEDLINE collection of abstracts of the US National Library of Medicine. To experiment with this set of documents, we asked our medical expert to select three sets of journals, ranging from very relevant to less relevant. The corresponding document corpora are denoted with $\mathcal{C}_1 \subset \mathcal{C}_2 \subset \mathcal{C}_3$, and contain respectively 5 367, 71 845, and 378 082 abstracts dated between January 1982 and November 2000. In the results section, the corpus $\mathcal{C}_3$ is used, unless stated otherwise.

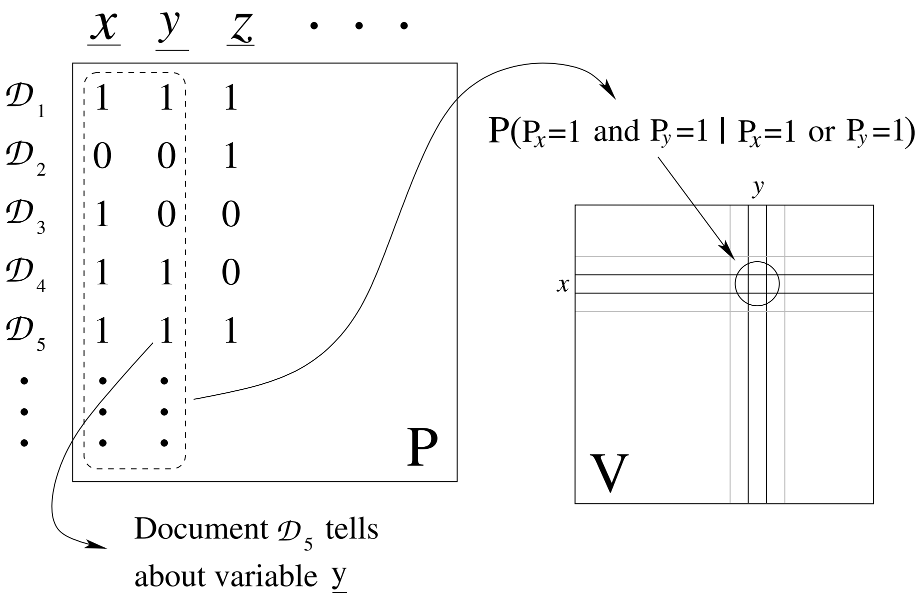

x
y
z
• • •
D1
D2
D3
D4
D5
1 1 1
0 0 1
1 0 0
1 1 0
1 1 1
P
P(Px=1 and Py=1 | Px=1 or Py=1)
y
x
V
Document D5 tells about variable y

Figure 7.4: A schematic overview of the binary presence approach to extract a relation matrix $\boldsymbol{V}$ from a corpus of electronic documents. The binary presence matrix $\boldsymbol{P}$ indicates *per document* which variables are mentioned in the document. Using this matrix, we can estimate the probability that a random document mentions *both* $\underline{x}$ and $\underline{y}$, given that this document mentions *either* $\underline{x}$ or $\underline{y}$. We use this probability as a measure of the relation between the variables $\underline{x}$ and $\underline{y}$.

The pairwise relation matrices obtained by the direct similarity technique is visualized in Figure 7.5, while the result of the binary presence technique is shown in Figure 7.6 using $\mathcal{C}_3$. Both the grey-scale of the edges and their thickness indicate the strength of the relation between the connected variables.

To evaluate at first glance the performance of these text-based variable relations and their potential for defining a Bayesian network structure prior, we compared them with the edge probabilities based on the data set. We compute the probability of seeing an edge between $\underline{x}$ and $\underline{y}$ by computing the probability of the Bayesian network structure containing only this specific edge. Because we removed constant multiplication factors in Equation 5.9, we divide the probability of the single-edge network structure with the probability of the network structure containing no edges. Doing so, the constant multiplication factors will be cancelled out. Because these network structures are very much alike, computing their probabilities involved almost the same computations; the ratio of the network probabilities can be simplified to the ratio of the probability of the local substructures $\underline{x} \rightarrow \underline{y}$ and $\phi \rightarrow \underline{y}$. We indicate with $\underline{x} \rightarrow \underline{y}$ that $\underline{y}$ has node $\underline{x}$ as parent, while $\phi \rightarrow \underline{y}$ indicates that $\underline{y}$ has no parents. We computed

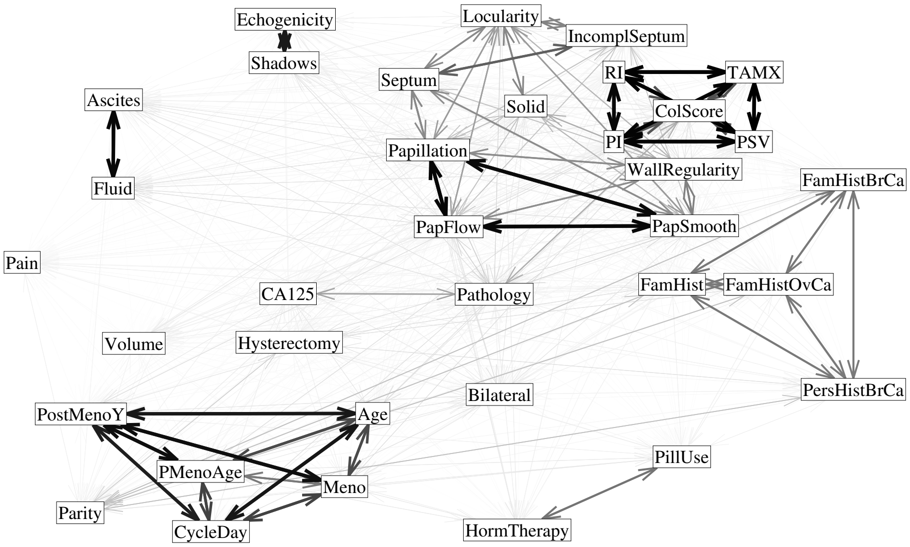


Figure 7.5: A visualization of the pairwise relationships obtained from the direct similarity technique. The strength of a relation is indicated by the grey-scale and thickness of the corresponding edge.

for each pair of variables $\underline{x}$ and $\underline{y}$, the logarithm of this ratio:

$$\begin{aligned}\boldsymbol{V}_{\underline{y}\underline{x}} &= \log\left(\frac{\mathrm{p}(\underline{x} \to \underline{y})}{\mathrm{p}(\phi \to \underline{y})}\right) \\ &= \log(L(\,\underline{y}\,|\,\underline{x}\,)) - \log(L(\,\underline{y}\,|\,\phi\,)).\end{aligned}$$

These are simply the local substructure probabilities where only one parent is present minus the local substructure with no parents. The relatedness matrix that is created in this way is shown in Figure 7.7. The four relatedness matrices (the expert-based, two text-based, and the data-based) show a good resemblance, considering their completely different origin.

Figure 7.8 displays a scatter plot for these edge probabilities; since our goal is the prediction of the *Pathology* variable, we plotted only the pairwise relations involving *Pathology* for the text-based and data-based scores. On the x axis are the text-based relations using the binary presence method based on $\mathcal{C}_3$, while the y axis displays the data-based relations. Although the relations are not on a straight diagonal line, they do correspond up to a certain degree. This is an indication that there at least comes something useful out of our text-mining techniques.

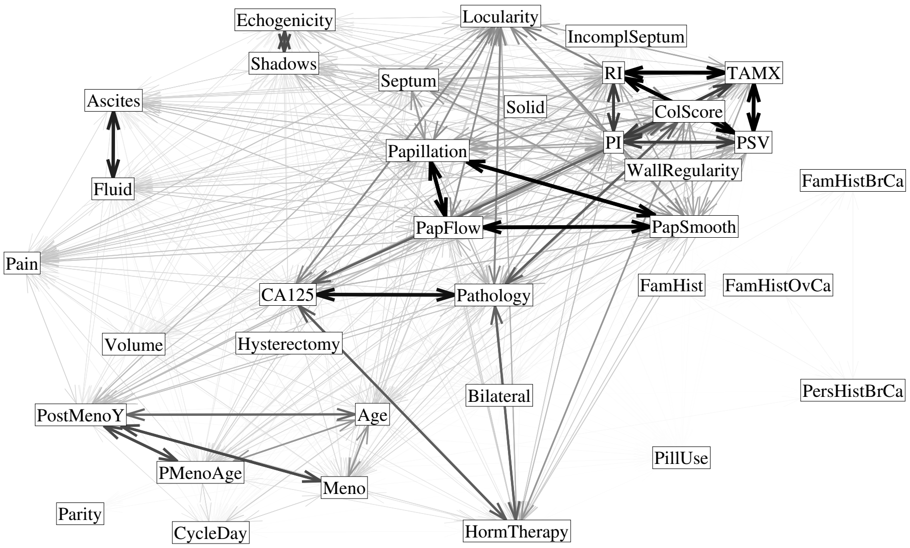


Figure 7.6: A visualization of the pairwise relationships obtained from the binary presence technique using $\mathcal{C}_3$. The strength of a relation is indicated by the grey-scale and thickness of the corresponding edge.

7.3 Learning curves and performance measure

We will describe a whole range of different classification models. To be able to compare the performance of these models, we will use the area under the receiver operating characteristics curve as the main performance measure because it is classification oriented, widely used to denote the performance of medical classification systems and has a nice and clear interpretation (see Section 3.3.2).

We are not only interested in the performance of our models, but would also like to visualize the learning behaviour of the different models. We will do this by generating *learning curves* for each model based on the ROC performance measure. Such a learning curve plots the ROC performance of a certain model computed on a test set with respect to the number of training samples used to learn the model. For this reason, we choose for a repeated two-fold cross-validation setup: for a certain percentage of training samples used (indicated on the x axis), we will divide the data set several times at random in a training set containing the required percentage of samples. The test set contains the remaining samples. Based on the training set, the a posteriori distribution for the model is computed. Finally, this a posteriori distribution is used to compute the area under the ROC curve for the samples in the test set. The average area under the ROC curve for these random train/test set divisions is indicated on the y axis.

Such a learning curve gives us a nice impression of the learning characteristics

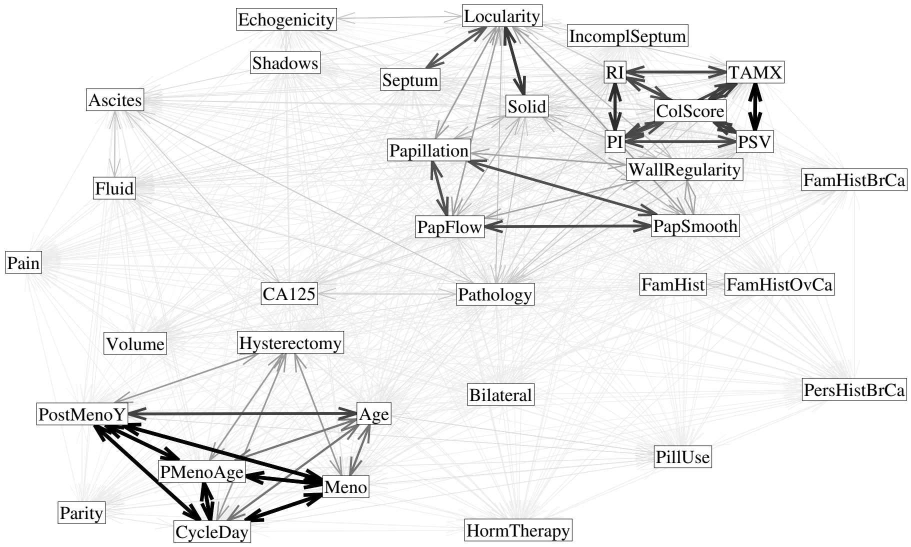


Figure 7.7: A visualization of the pairwise relationships obtained from the data using the logarithm of the local substructure probabilities. The strength of a relation is indicated by the grey-scale and thickness of the corresponding edge.

of each model. The further we go on the x axis, the more training samples are used to learn the a posteriori distribution. Since more samples could be used to learn the model, its performance should increase similarly.

Although these learning curves *look* like ROC curves, they are not. A learning curve is constructed by repeatedly computing the area under the ROC curve for models using different training sets. The learning curves we will present are computed for values on the x axis that range from 0 to 0.9. A value of zero indicates that no training samples are used to compute the posterior distribution and that all the samples are used for computing the area under the ROC curve, which is plotted on the y axis. A value equal to 0.9 indicates that 90% of the samples are used to define the a posteriori distribution, leaving only 10% of the samples to compute the area under the ROC curve. We will not use more than 90% of the samples for training because we want to have at least 10% of the samples to estimate the performance.

7.4 Results

In this section, we will present an overview of the performance of the different models that are discussed in this thesis. We will use the learning curves introduced in the previous section to compare the learning behaviour of these different models.

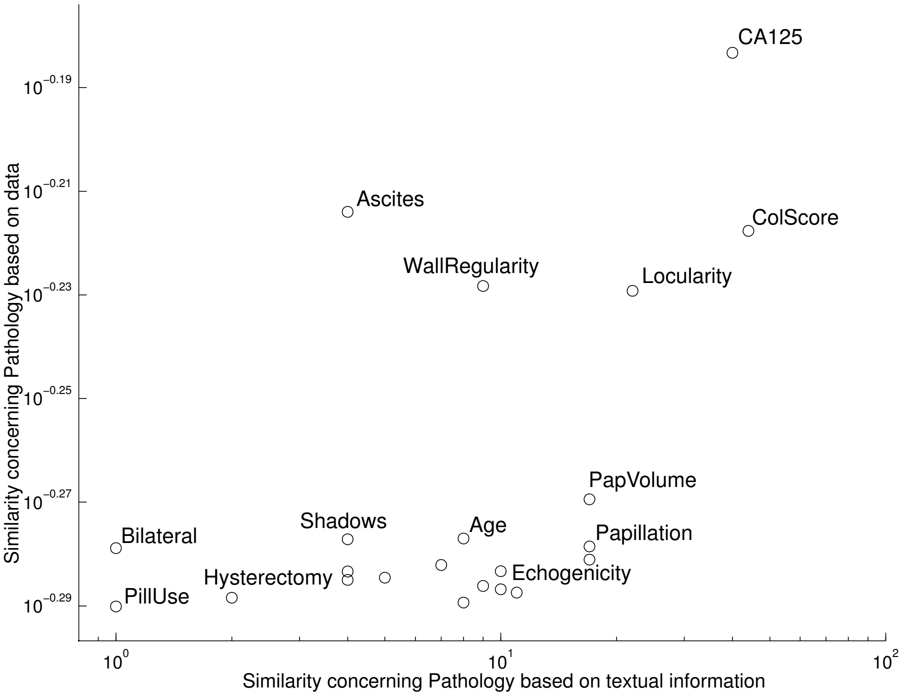


Figure 7.8: A scatter plot comparing pairwise relation based on textual information (x axis) and data records (y axis). For the text-based relations, the binary presence technique was applied using corpus $\mathcal{C}_3$. Only relations concerning *Pathology* are considered for clarity.

7.4.1 Previous results

The first predictive models used to classify ovarian tumours were based on single variables, such as *CA125* or risk indices such as the *risk of malignancy index* (RMI) [46, 84]. This last one is a well studied method to predict the malignancy of an ovarian mass. It is a multiplicative rule that takes into account *ultrasound* information *(Ascites*, *Locularity* and *Bilaterality)* together with the *Menopausal* status and the level of *CA125*.

$$\begin{array}{rcl} \text{RMI} & = & \mathit{Ultrasound} \times \mathit{Meno} \times \mathit{CA125} \\ \mathit{Ultrasound} = 0 & \text{if} & \mathit{Morph} = 0 \\ \mathit{Ultrasound} = 1 & \text{if} & 0 < \mathit{Morph} \leq 1 \\ \mathit{Ultrasound} = 3 & \text{if} & 1 < \mathit{Morph} \\ \mathit{Morph} & = & \mathit{Asictes} + \mathit{Unisolid} + \mathit{Multi} + 2\mathit{Multisolid} \\ & & \mathit{Solid} + \mathit{Bilateral} \end{array}$$

The simplicity of this rule, its robustness when tested prospectively, together with its reasonable performance with an area under the ROC curve of 0.8891462 makes this a widely used rule of thumb to triage women into different risk categories.

Table 7.9 shows the area under the ROC curve performance for univariate models. As such, the first column in this table says that if we use the variable *CA125* as the test function $T(\underline{x} \mid \phi)$ to compute the area under the ROC curve, we get a value of 0.8323409. The empty set symbol ϕ indicates that there are no parameters present in the univariate *CA125* model, we simply use the corresponding *CA125* value. Variables that are negatively correlated to *Pathology* were reversed so they result also in an area under the ROC curve above 0.5.

Variable	Area under ROC
CA125	0.8323409
Locularity	0.8170526
Colour Score	0.7910523
TAMXV	0.7211011
PSV	0.7072227
Solid	0.7482750
Ascites	0.7419108
Fluid	0.7339950
Wall Regularity	0.7261500
Volume	0.6912807
Age	0.6802379
Papillation Flow	0.6739682
PI	0.6727946
Meno	0.6632719
Papillation Smoothness	0.6625000
Septum	0.6607593
Papillation	0.6428875
RI	0.6382168
Bilateral	0.6107672

Figure 7.9: The area under the ROC curve performance of the most important variables to predict tumour malignancy.

Standard statistical studies indicate that a multi-variate approach — the combination of several variables — is necessary for the discrimination between benign and malignant tumours, which is our motivation to apply both Bayesian networks and neural networks to this classification task.

7.4.2 Bayesian networks

The Bayesian networks that we will use for our experiments all use discrete variables. This requires that we discretize the continuous variables.

Discretization

Together with the medical expert, we defined sensible discretization bins for the continuous variables, as presented in the Appendix A. The second column contains these intervals. A certain observation will be converted to the corresponding value of the first column according to the interval it belongs to. Although the primary usage of these bins is to discretize the continuous variables to use them for Bayesian network learning, we will also use them to convert the discrete virtual data sets generated from the prior Bayesian network into continuous data sets to estimate the informative prior for the multilayer perceptron model.

When discretizing the continuous variables, some information is inherently lost. Table 7.10 gives an impression of this loss of information by displaying the area under the ROC curve performance for univariate models containing respectively the original and the discretized variables. As can be seen from this table, the loss of information results often in a worse performance, although some variables like *TAMXV* or *Parity* seem to benefit from the discretization.

Variable	Continuous	Discretized
CA125	0.8323409	0.7939666
TAMXV	0.7211011	0.7786074
Fluid	0.7339950	0.7077820
Age	0.6802379	0.6685570
Volume	0.6912807	0.6576953
PI	0.6727946	0.6139808
Parity	0.5887917	0.5892328
RI	0.6382168	0.5653119
Pill Use	0.5153355	0.5282530
Septum	0.5167297	0.5275992

Figure 7.10: The area under the ROC curve performance of the continuous variables and their discretized versions.

Learning with non-informative priors

Once we have discretized the continuous variables, we can perform Bayesian network learning. Ideally, learning a Bayesian network model in the Bayesian framework consists of finding the joint posterior distribution over both the network structures and the parameters for this network structure:

$$\mathrm{p}(\,\mathcal{S}_{\mathrm{bn}}, \boldsymbol{\theta} \,|\, \mathbf{D}, \xi\,) = \mathrm{p}(\,\mathcal{S}_{\mathrm{bn}} \,|\, \mathbf{D}, \xi\,)\, \mathrm{p}(\,\boldsymbol{\theta} \,|\, \mathcal{S}_{\mathrm{bn}}, \mathbf{D}, \xi\,).$$

This posterior distribution can be used to compute the probability of malignancy given the observation $\boldsymbol{x}$ by marginalizing over both the network structures and their parameters:

$$\mathrm{p}(\,\mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x},\mathbf{D},\xi\,) = \sum_{\mathcal{S}_{\mathrm{bn}}} \mathrm{p}(\,\mathcal{S}_{\mathrm{bn}}\,|\,\mathbf{D},\xi\,) \int_{\boldsymbol{\Theta}} \mathrm{p}(\,\mathcal{C}_{\mathrm{P}}\,|\,\boldsymbol{x},\mathcal{S}_{\mathrm{bn}},\boldsymbol{\theta}\,)\,\mathrm{p}(\,\boldsymbol{\theta}\,|\,\mathcal{S}_{\mathrm{bn}},\mathbf{D},\xi\,)\,d\boldsymbol{\theta}.$$

The resulting malignancy probabilities for the different records in the test set are then used to compute the area under the ROC curve performance.

We approximate the summation over the network structures by finding a set of network structures with a high posterior probability and estimate the integral with a Monte Carlo summation. Finding the set of network structures involves structure learning, which is explained in Section 5.2.1. The Monte Carlo sum can easily be generated because the parameter posterior $\mathrm{p}(\,\boldsymbol{\theta}\,|\,\mathcal{S}_{\mathrm{bn}},\mathbf{D},\xi\,)$ is a known Dirichlet distribution.

A typical network structure from the posterior structure distribution can be seen on Figure 7.11. This structure was learned using all the data samples. The links in this structure make more or less sense from a medical point of view. Note that the direction of a link between two variables does not necessarily indicate a causal relation between these variables.

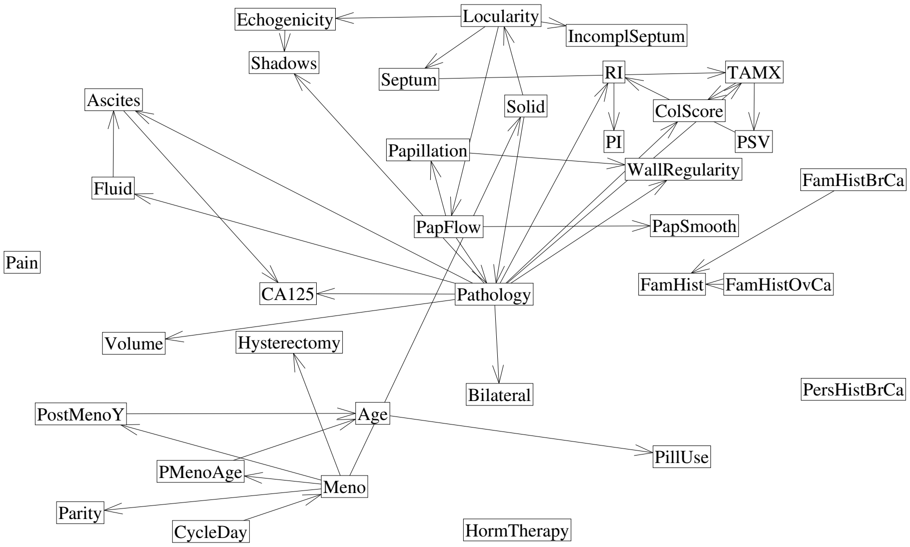


Figure 7.11: A typical network structure from the posterior structure distribution for the ovarian tumour problem.

We used these computed class probability predictions to compute a learning curve based on the area under the ROC curve performance measure (see Section 7.3). A set of 200 high probability network structures was searched for each of the 1000 two-fold cross-validation session. For each network structure, the Monte Carlo summation was approximated with 200 terms. Figure 7.12

shows the learning curve; the proportion of training samples used is indicated on the x axis, while the mean area under the ROC curve performance is indicated on the y axis. The learning curve shows the expected behaviour. When almost no samples are used to learn the model, the performance is the same as a random classifier, which reflects in an area under the ROC curve performance around 0.5. When more and more samples are used to compute the posterior, the model learns and the performance climbs to an area under the ROC curve of 0.941.

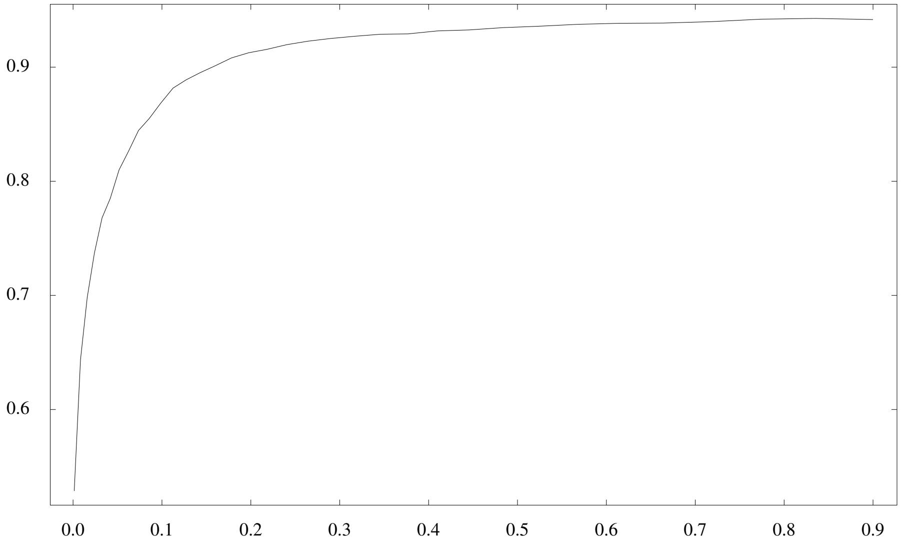


Figure 7.12: The learning curve for the Bayesian network model. Both the structure and parameter prior are non-informative.

We can improve this learning behaviour in three different areas. At first, it would be nice if the model could learn *faster*. This means that we want the learning curve to climb more steeply in the small sample range (the left part of the graph). Next, we would like the model to start from a *reasonable* value, instead of starting from the same performance as a random classifier. At last, it would also be nice if our model could achieve a *higher* performance in the large sample range (the right part of the graph).

Expert and textual structure prior

We can achieve the first goal, the improved learning characteristics in the small sample range, by using a *structure* prior for the Bayesian network model. If this prior distribution over the network structures is reasonably correct, a correct structure will be learned using fewer data records. This correct structure will in its turn enable the model to learn how the actual dependencies look like (the parameters of the Bayesian network) using fewer data records.

We have two main sources of Bayesian network structure priors. At first, we can use the relatedness matrix that was defined by the expert, as explained in Section 7.2.2, and use it to define a probability distribution over the Bayesian network structure space using the method introduced in Section 5.5.1. Second, we can use one of the relatedness matrices that were computed from the domain literature, as explained in Section 7.2.3.

Before we compare the learning curves corresponding to these different structure priors, we have to deal with the single parameter ν that determines how a structure prior is constructed from a relatedness matrix: in Section 5.5.1, we introduced this parameter to control the density of the networks that result from the distribution. Figure 7.13 shows the influence of this parameter ν on the learning curve, using the expert structure prior. We computed learning curves for the values $\nu \in \{2, 1, 2^{-1}, 2^{-3}, 2^{-5}\}$. If ν is chosen too small, the structure prior is restricting the learning a lot and it takes many samples before the information contained in these samples is able to cancel out the wrongly specified prior. If we take ν larger, the Bayesian network model learns faster. From Figure 7.13, we can see that there is no substantial difference anymore once ν becomes larger than 1. From now on, we fix the parameter ν to a value of 2, as this gives a nice performance and seems reasonable.

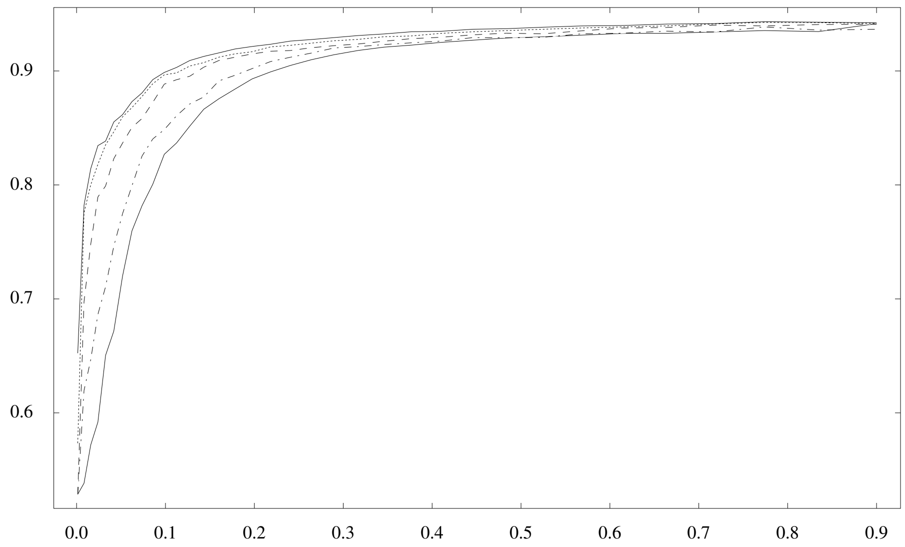


Figure 7.13: The different learning curves corresponding to the Bayesian network model using the expert structure prior for different values of ν: the bottom full line uses $\nu = 2^{-5}$, the dashdotted line uses $\nu = 2^{-3}$, the dashed line corresponds to $\nu = 2^{-1}$, the dotted line corresponds to $\nu = 1$, and the upper full line finally uses $\nu = 2$.

Figure 7.14 displays the learning curves of four Bayesian network models.

Each model uses a different structure prior: the expert structure prior or one of the three text-based structure priors, as described in Section 7.2.3.

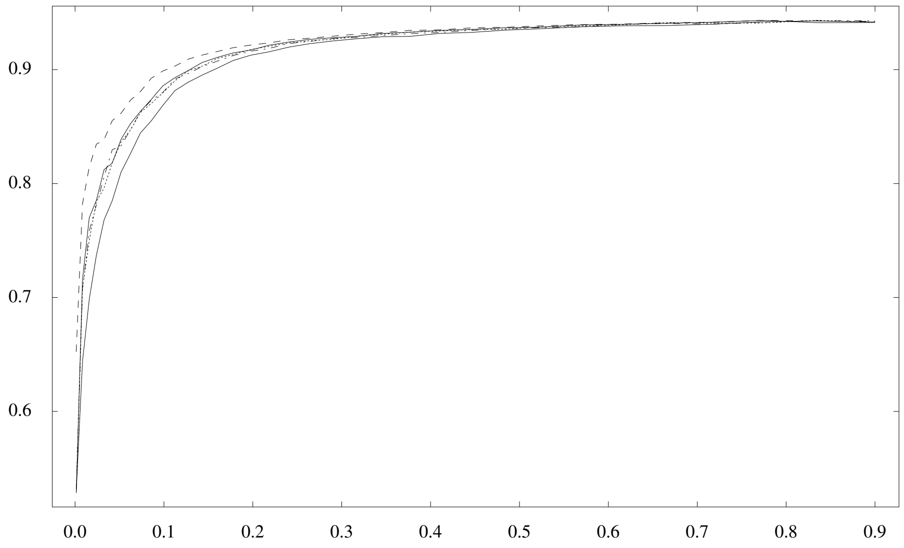


Figure 7.14: The different learning curves correspond to the Bayesian network model using different structure priors and a density value of $\nu = 2$: the dashed line corresponds to the expert prior, the dotted and the full line correspond respectively to the binary presence-based prior using the document corpora $\mathcal{C}_1$ and $\mathcal{C}_3$. The dash-dotted line corresponds to the direct similarity-based prior. Finally, the second full line with the worst performance corresponds to a model using no structure prior.

The expert prior results in the best performance, followed by the binary presence method to extract structural information from the literature. The direct similarity has a worse performance, but is still better than the model using no structure prior. The differences between these priors are gradually cancelled out if more training samples are used. This is the behaviour that we expected.

Expert Bayesian network

The structure prior used in the previous section successfully incorporates information about the domain, but is restricted to *which* variables interact with each other. This has an effect on how fast the model learns. To accomplish the second requirement described at the end of Section 7.4.2 — creating a model that performs better than a random one even when no data samples are observed — we have to incorporate information about *how* the different domain variables interact with each other. We tried to achieve this goal by asking our expert to construct a Bayesian network structure *and* a corresponding parametrization.

This network structure is shown on Figure 7.1. We gave this prior Bayesian network an equivalent sample size of 50 (see Section 5.3.1).

We computed the learning curve for this *fixed structure* Bayesian network and displayed it on Figure 7.15 using a full line. To be able to compare this curve easily with the previous curves, we indicated also the Bayesian network model using the expert structure prior (dashed line) and no prior information (dotted line).

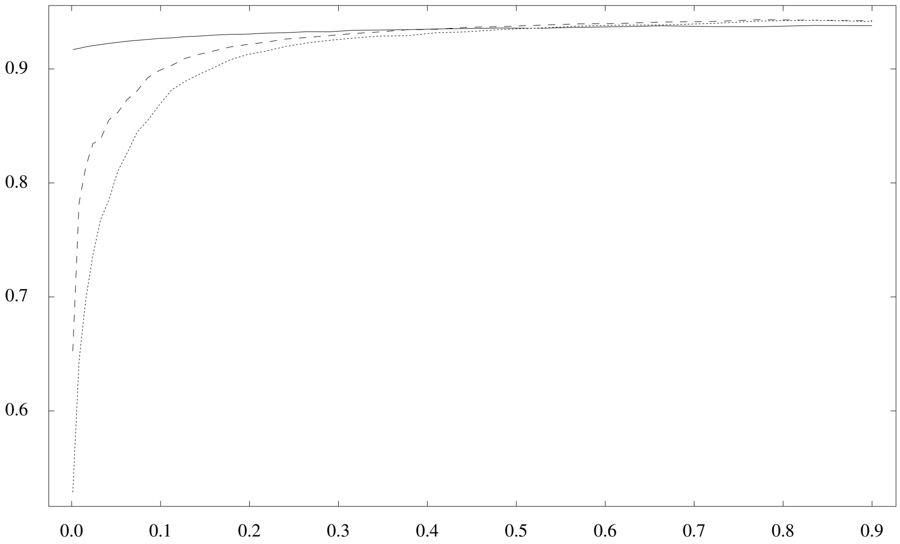


Figure 7.15: The learning curve corresponding to the fixed Bayesian network structure and parameters specified by Prof. Timmerman (full line). The model using no prior information is indicated using a dotted line, while the model using the expert structure prior is given by the dashed line.

The learning curve for the fixed structure Bayesian network indeed starts at a good performance level of 0.917, even without any data observed, which is a lot better than the risk of malignancy index or any univariate model. Unfortunately, the fixed structure of this model and the restricted subset of variables it contains, limits its performance somewhat on the long run; in the large sample range, its performance is worse than any of the models that have an unrestricted structure.

The third requirement — achieving a higher performance in the large sample range — will be dealt with by transforming the information representation to another model class that has better learning capabilities, as explained in the following sections.

7.4.3 Logistic regression

A first alternative model we will look at, is logistic regression. These models are an extension of the well-known linear regression, with an eye towards classification (see Section 6.2.3).

Non-informative prior

We start with the basic logistic regression model, using only a complexity-based prior. Before we start the computations, we perform the necessary pre-processing steps, as explained in Section 7.2.1. Based on this pre-processed data set and a complexity-based prior consisting of a multivariate Gaussian with zero mean and 0.07 variance, we can specify the a posteriori distribution. The variance for this prior was chosen so low to anticipate the overtraining that otherwise occurs when only a few samples are present in the training set. This small variance did not have any negative effect on the performance when more samples are used to define the posterior.

Using the hybrid Monte Carlo Markov chain method, we sampled 100 parametrizations from this a posteriori distribution, which were used to approximate the Bayesian malignancy predictions needed to compute the area under the ROC curve. We used 100 leapfrog steps for the hybrid Monte Carlo Markov chain method, and estimated the step size using the procedure described in Section 6.7.2. We initialized the Markov chain with the maximum a posteriori parameter vector that we learned using the scaled conjugate gradient algorithm (see Section 6.6.2). This process was repeated for 1000 two-fold cross-validation sessions to get an estimate of the mean of the performance of the model.

Figure 7.16 displays the learning curve for this model (full line), together with the Bayesian network without any prior information (dotted line) and the Bayesian network using the expert structure prior (dashed line).

The logistic regression model has much better learning characteristics in the small sample range than the Bayesian network. This has to do with the large number of parameters that have to be learned in the Bayesian network case and the information that is lost by discretizing the continuous variables [32]. The performance of the logistic regression model is also slightly better in the large sample range.

Informative prior

Just as we required for the Bayesian network, we want our logistic regression model to start with a reasonable performance, even if no samples are observed. Unfortunately, it is hard for an expert to specify reasonably good values for the logistic regression coefficients directly. To deal with this problem, we designed the transformation procedure that was introduced in Chapter 4: we will use the expert Bayesian network from Section 7.4.2 and transform this information to an informative a priori distribution over the logistic regression parameter space. Using the terms of Chapter 4, the expert Bayesian network is the *donor* model, while the logistic regression model is the *acceptor* model.

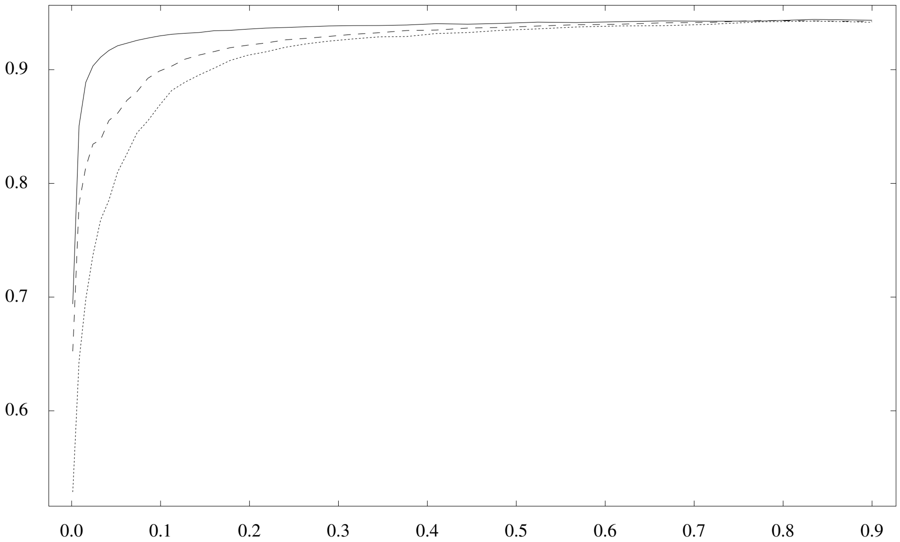


Figure 7.16: The learning curve for the logistic regression model using a complexity-based prior (full line). The Bayesian network models using the expert structure prior (dashed line) and no prior information (dotted line) are also indicated for comparison.

We used the procedure described in Section 4.4 to generate a set of logistic regression parametrizations from this a priori distribution. Using these parametrizations, we estimated the informative a priori distribution using a multivariate Gaussian distribution.

The virtual data sets we used to generate sample vectors from the informative logistic regression parameter distribution contained 1000 records. Each discrete record generated from the expert Bayesian network was transformed into a continuous record using the procedure described in Section 4.4.2 and the discretization intervals that are given in Appendix A. Next, each continuous record has undergone the same pre-processing steps that were applied to the continuous real data set before it was used to learn the complexity-based logistic regression from the previous section.

We used the hybrid Monte Carlo Markov chain method to generate a logistic regression parametrization from the posterior based on the continuous virtual data set. The settings for the hybrid Monte Carlo Markov chain method were the same as those used for the complexity-based logistic regression model.

Once the informative logistic regression a priori distribution is estimated, it can be used to compute a learning curve. Again, the hybrid Monte Carlo Markov chain method was used to approximate the Bayesian malignancy probability that is needed to compute the area under the ROC curve. The settings of this method are again the same as previously.

The learning curve corresponding to this informative logistic regression model (full line) is displayed on Figure 7.17, together with the complexity-based logistic regression model (dashed line) and the expert Bayesian network (dotted line).

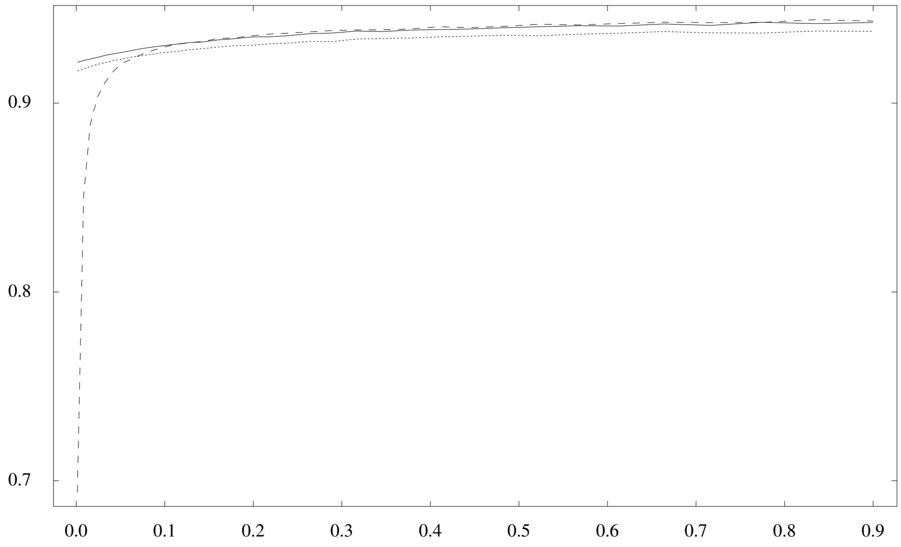


Figure 7.17: The learning curve of the informative logistic regression model (full line), together with the expert Bayesian network performance (dotted line) and the complexity-based logistic regression model (dashed line).

Although some information has been lost during the transformation, the informative logistic regression performs better than the expert Bayesian network if no samples are used. This is a coincidence and is certainly not a general behaviour of the transformation technique. We also see that the logistic regression model still learns, as its performance increases if more samples are used to specify the a posteriori distribution.

7.4.4 Multilayer perceptrons

The last model class we will use, is the class of multilayer perceptrons. These models are an extension of logistic regression models and can have nonlinear classification boundaries and are introduced in Chapter 6.

Non-informative prior

We start with the complexity-based multilayer perceptron model. We chose to use one hidden layer, containing two hidden neurons as the network structure. The additional model complexity that is introduced using more hidden neurons is not necessary for the ovarian tumour problem.

Similar to the logistic regression case, the complexity-based prior consists of a multivariate Gaussian with zero mean. We varied the variance depending of the type of the weight: we used a variance of 0.15 for the weights connecting the input to the hidden layer and the hidden layer to the output. The variance of the input bias was set to 5.0 to allow translations of the input variables. The bias for the output bias was set to 0.5. After pre-processing the data set in exactly the same way as for the logistic regression case, we also used the hybrid Monte Carlo Markov chain method to approximate the Bayesian malignancy probabilities used to compute the area under the ROC curve.

Figure 7.18 shows the learning curve for this model (full line), together with the complexity-based regression model (dotted line).

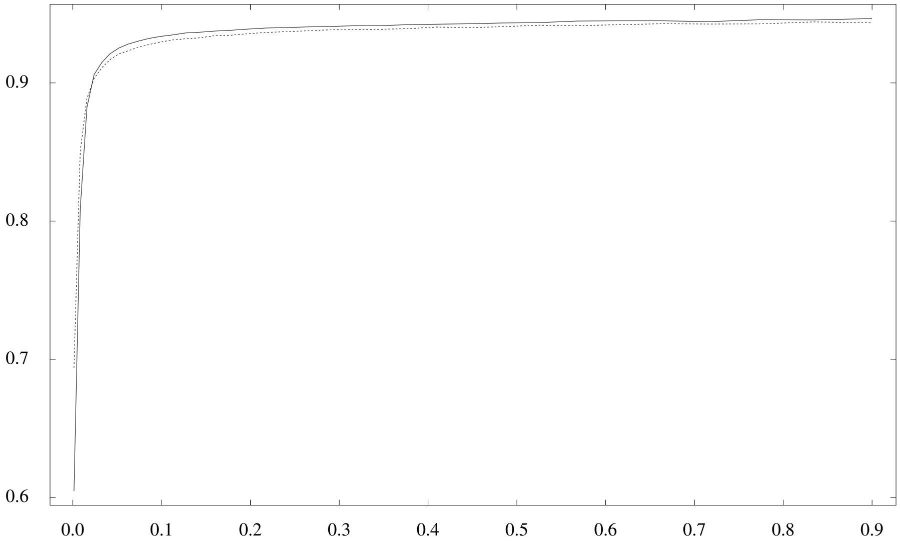


Figure 7.18: The learning curve for the complexity-based multilayer perceptron model (full line), together with the complexity-based logistic regression model (dotted line).

The multilayer perceptron model has almost the same learning behaviour as the logistic regression model. It learns a bit more slowly in the very small sample range, because it has more parameters. On the other hand, it is a more flexible model than the logistic regression model, which results in a better performance once some data samples are observed.

Informative prior

Finally, we transformed the information from the expert Bayesian network to a prior distribution over the neural network weight space. The steps to generate sample parametrizations from this informative distribution are exactly the same as for the logistic regression case. Only the estimation of the distribution becomes more complicated because of the symmetries that are present in neural network weight space when one or more hidden layers are present (see Section 6.9). Again, we used the hybrid Monte Carlo Markov chain to approximate the Bayesian predictions that are necessary to compute the learning curves.

Figure 7.19 displays the learning curve for the informative multilayer perceptron model (full line), together with the informative logistic regression model (dotted line) and the expert Bayesian network model (dashed line). Please note that the scale of this figure is substantially different from the previous figures, because there is no model present that has a performance around 0.5 if no records are used to specify the a posteriori distribution.

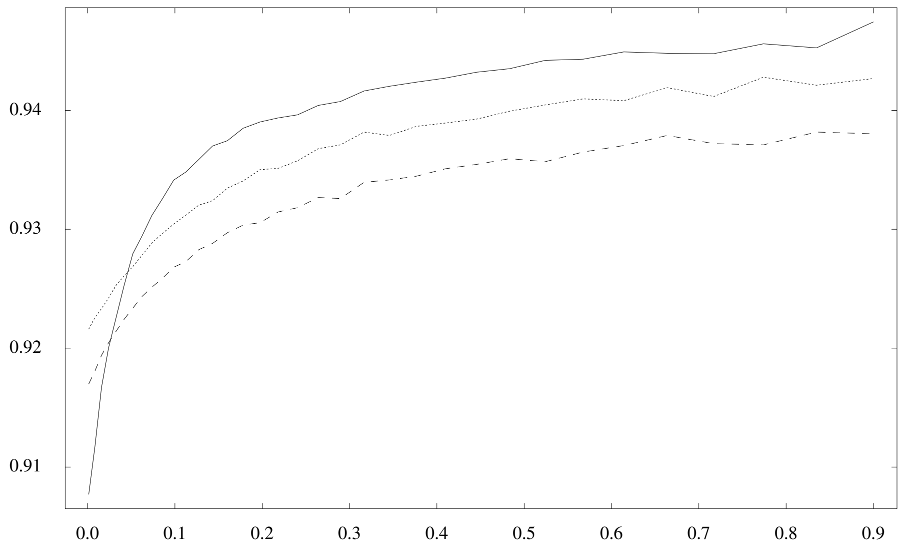


Figure 7.19: The informative multilayer perceptron model (full line), together with the informative logistic regression model (dotted line) and the expert Bayesian network (dashed line).

The informative multilayer perceptron model starts with a performance that is lower than both the expert Bayesian network model and the informative logistic regression model. But because it is more flexible than these two models, it achieves the best performance already after observing a few samples.

We have presented a whole range of models and their corresponding learning curves. This gave a visual impression of the learning characteristics and performance of the different models. We will end this result section with a multi-sample *t*-test to find out if the performance differences between the models is significant [72]. We will do this test for the 5%-95% train-test division and for the 70%-30% train-test division. Table 7.20 summarizes the statistics for the first case, while Table 7.21 summarizes the statistics for the latter. These tables present the sample mean, the sample standard deviation and the 95% confidence interval for the mean based on the pooled standard deviation.

Although the variances of the area under the ROC curve values for the different cross-validation sessions are not the same for the different models, we can still use the *t*-test if the sample sizes are the same [13]. We have control over the number of cross-validation sessions that we performed and chose them to be the same for the different models (1000).

Model	Average	Std	95% CI for mean
BN noninformative	0.8100	5.02e-03	[0.8079, 0.8121]
BN expert structure prior	0.8617	2.65e-03	[0.8596, 0.8638]
expert BN	0.9234	9.49e-06	[0.9213, 0.9255]
LR noninformative	0.9210	1.30e-04	[0.9189, 0.9231]
LR informative	0.9269	1.96e-05	[0.9248, 0.9290]
MLP noninformative	0.9252	1.27e-04	[0.9231, 0.9273]
MLP informative	0.9279	7.04e-05	[0.9258, 0.9300]

Figure 7.20: A summary of the statistics of the performance of the different models for a 5%-95% train-test division. The first column indicates the model, the second gives the sample mean of the area under the ROC curve for the different cross-validation session, the third column gives the sample standard deviation and the last column presents the 95% confidence interval for the mean, based on the pooled standard deviation.

Model	Average	Std	95% CI for mean
BN noninformative	0.9399	2.35e-04	[0.9389, 0.9408]
BN expert structure prior	0.9416	2.12e-04	[0.9407, 0.9425]
expert BN	0.9372	2.04e-04	[0.9363, 0.9381]
LR noninformative	0.9426	2.22e-04	[0.9417, 0.9435]
LR informative	0.9412	2.23e-04	[0.9403, 0.9421]
MLP noninformative	0.9444	2.11e-04	[0.9435, 0.9453]
MLP informative	0.9448	2.08e-04	[0.9439, 0.9457]

Figure 7.21: A summary of the statistics of the performance of the different models for a 70%-30% train-test division. The first column indicates the model, the second gives the sample mean of the area under the ROC curve for the different cross-validation session, the third column gives the sample standard deviation and the last column presents the 95% confidence interval for the mean, based on the pooled standard deviation.

From these summary statistics, we compute that there is convincing evidence that the multilayer perceptron model with informative prior performs better than the expert Bayesian network (one sided *p*-value of 0.0025) the noninformative multilayer perceptron model (one sided *p*-value of 0.05) for the 5%-95% train-test division. There is inconclusive indication that this multilayer perceptron model with informative prior performs better than the logistic regression model with informative prior (one sided p-value of 0.25).

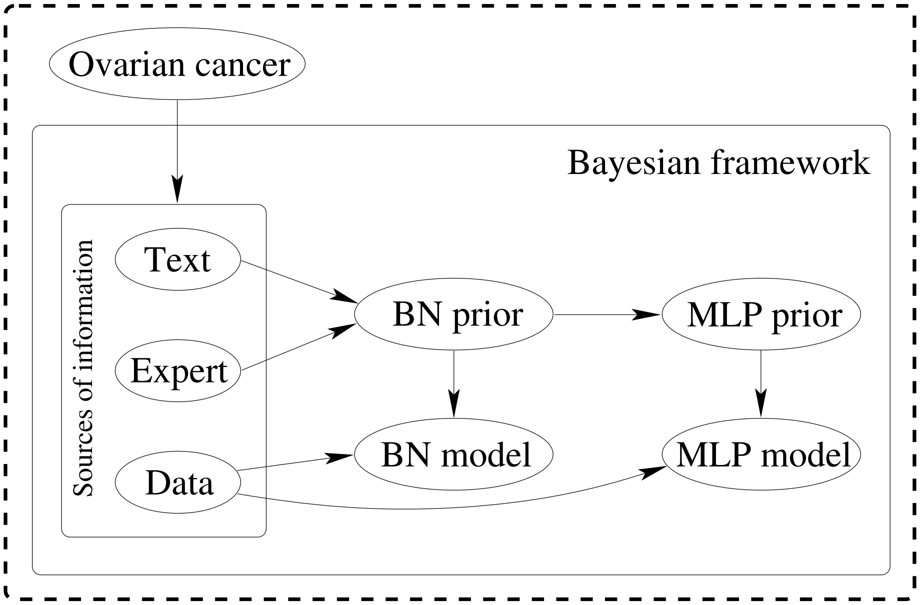
In exactly the same way, we can conclude that there is a significant difference for the 70%-30% train-test division between the multilayer perceptron model with informative prior and the informative logistic regression model (one sided p-value smaller than 0.0005) and the Bayesian network model with the expert structure prior (one sided p-value smaller than 0.0005). This time there is inconclusive indication that the informative multilayer perceptron model performs better than the complexity based multilayer perceptron model (one sided p-value of around 0.25).

7.5 To conclude

A lot of research was already performed on the field of ovarian tumours. We gave a brief overview of the most common classification systems that were applied in the passed and contributed to this research by designing and implementing a method to combine both expert and textual information with statistical data. This included extracting the prior expert information using Bayesian networks and transforming this information to a multilayer perceptron for efficient learning from data. This data was pre-processed and investigated using standard techniques and logit plots. We compared the performance of various models using the area under the ROC curve and provided insight into their learning characteristics using learning curves.

Chapter 8

Implementation



The methods and techniques discussed so far require a great deal of computation and a large number of different algorithms have to be combined. As a consequence, implementing *these techniques is not a trivial task.*

8.1 General setup

Although software packages exist for some of the main building blocks of this thesis, there are only few and most of them are too restricted to use in practice. The most important are the "Bayesian Network Toolbox", developed by Kevin

Murphy (`http://www.ai.mit.edu/∼murphyk/`), and the "Open Probabilistic Network Library" (`https://sourceforge.net/projects/openpnl/`). Radford Neal has developed the software package "Software for Flexible Bayesian Modelling" (`http://www.cs.toronto.edu/∼radford/fbm.software.html`), which can perform Bayesian computations.

After a lot of consideration and lessons learned from previous implementations, we decided to implement *everything* from scratch. Although this seems a strange choice, this guarantees a complete understanding of what is happening, a complete independence from other code, a maximum flexibility, and an optimal efficiency, both computational as memory related. From scratch means: from pseudo-random number generation and basic data structures over distributed computations up to inference and graphical output generation.

8.2 Language and development tools

The programming language that was used is C++ [2, 79, 24]. The well-known benefits of this language are its speed and rich object-oriented features, such as multiple virtual inheritance and virtual functions. Although its manual memory management is generally experienced as obstructing the development, it is crucial if efficient memory usage is necessary. As this is the case for Bayesian networks and multilayer perceptrons, C++-style memory management was a benefit for us. Most compilers can compile standard C++ and most libraries exist for different platforms, making the language more or less platform independent if certain programming rules are respected.

As development system, we chose for the GNU/Linux operating system for its stability, security, speed, open source features, and development friendliness. The compiler of choice was `gcc-3.2.3` in combination with `autoconf` and `automake`. The code was written using `emacs`. Debugging tools that proved useful are `valgrind`, a fantastic memory debugger, which is invaluable when working with complicated memory structures, `gdb`, a general purpose debugger, `ddd`, a graphical debugger and `gprof`, a profiling tool for optimizing the code. Although the main focus was to develop an API, a basic user interface was also built using `gtk2`.

Initially, Matlab was attractive as an alternative programming language because of its development speed and usage of matrices as the main building block. But its poor object-oriented approach and string handling, and its usage of a license server restricted its usage too much.

The source code contains 83 319 lines of code, 148 classes, 222 source and header files, and uses 2 450 647 bytes of space.

8.3 Main building blocks

The relations between the different techniques and algorithms are reflected in the inheritance structure of the software, where each basic procedure or object is represented by a class.

8.3.1 Graphical layout of the key classes

The main classes, their relation with each other and inheritance structure is displayed in Figure 8.1. A full line indicates a class inheritance relation, while the dotted lines indicate the important class membership relations. Class inheritance occurs when one class inherits all the methods and members from another class, usually to implement a similar but more specific class. As an example, *CPD* represents a general conditional probability distribution. *Dirichlet* inherits all the methods from *CPD* but implements a specific type of conditional probability distribution, a Dirichlet distribution. A membership relation indicates that a certain class has objects from another class as member variables (e.g., a *DataBlock* consists of several *Samples*). There are a lot of classes not indicated on this figure, because they are support classes, such as arrays, doubly linked lists, trees, matrices, network sockets, or graphical objects.

8.3.2 Data

One of the most central classes is the *DataBlock* class, which is the data structure that holds a data set.

It is basically an array of *Samples* together with a field name array. Each sample holds one data record, which can be either discrete or continuous. The values for these records can be either observed or missing. Each sample has some attributes that indicate if it should be used during training, or testing. The array contained in each sample can be divided into an *input* and an *output* part to be used as the input and target pattern for multilayer perceptron learning. The discretization bins to convert the continuous samples to discrete samples, are also stored in the *DataBlock* class, together with the settings to perform the reverse operation.

The important member functions include reading and writing from file and the discretization and continuation of a data set.

8.3.3 Random variables

To build up the probabilistic models discussed in this thesis, a general framework for dealing with random variables *(RV)* and conditional probability distributions *(CPD)* was developed. Both are abstract classes, meaning that the actual implementation of their member functions is left for the derived classes. The derived classes for *RV* include *RVC*, *RVD*, *RVectorC*, *RVectorD* and *RVMatrix*, which respectively implement continuous and discrete random variables, continuous and discrete random vectors and a random matrix. The derived classes from *CPD* represent the different conditional distributions, among which we have implemented uniform, linear-Gaussian, table, Dirichlet, Laplace, exponential, mixture of Gaussian, and Gamma distributions.

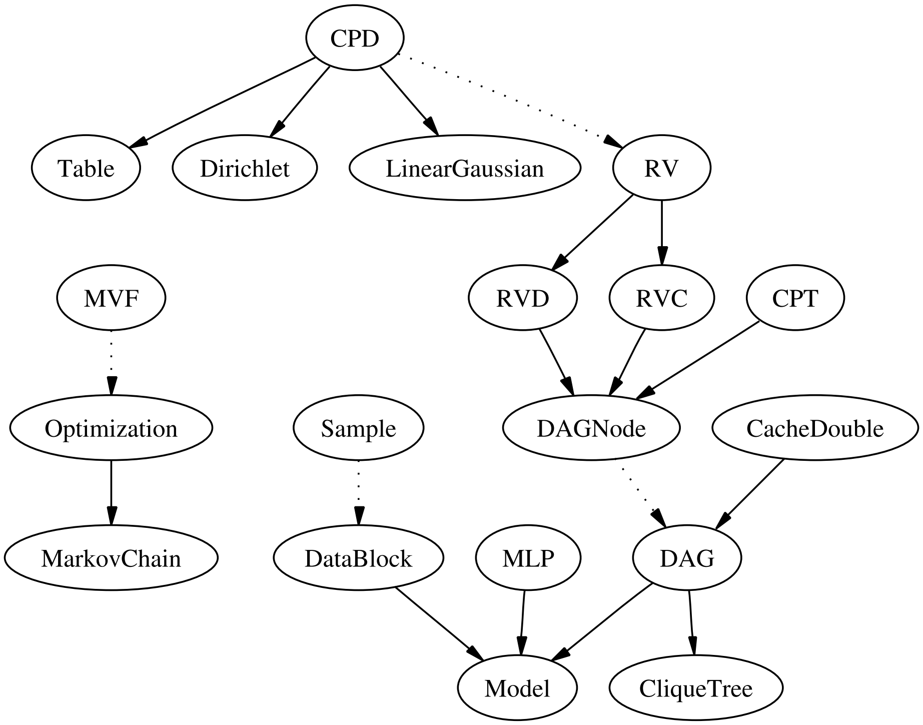


Figure 8.1: The structure of the main classes. A full line indicates class inheritance, a dotted line indicates class membership. *CPD* stands for conditional probability distribution, *RV* stands for random variable (discrete or continuous), *CPT* stands for conditional probability table, *MVF* stands for multivariate function, *MLP* stands for multilayer perceptron, and *DAG* stands for directed acyclic graph.

Each conditional distribution can have a set of *conditions* and *parameters*. The conditions are the random variables that are behind the condition sign | and are stored as an array of pointers to *RV* objects. The parameters of a conditional distribution is everything else that determines the distribution. Although we often think of parameters as being fixed, we represent them also with an array of pointers to *RV* objects. In this way, our parameters are random variables, and can have a distribution themselves. This allows us to work in the Bayesian framework in a natural way.

Each random variable has a pointer to a *CPD* object. If the pointer is not set, our random variable has a fixed distribution, otherwise we use the distribution represented by this object.

By combining these two objects, we can represent all the distributions we need to represent. As an example, we can represent the local dependency model of Figure 8.2 by using *Table* and *Dirichlet* distributions and binary variables as is indicated on Figure 8.3. *Par1* up to *par4* are the parameters of the *Table* distribution, the probabilities for $\underline{a}$ for the different configurations of the parents $\underline{b}$ and $\underline{c}$, while *hyper1* up to *hyper4* are the corresponding hyper-parameters for the *Dirichlet* distributions. The conditions for these Dirichlet distributions are left out as there are no conditions for this distribution.

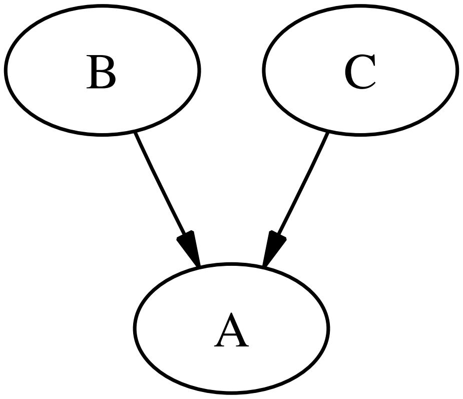


Figure 8.2: A local dependency structure involving two parents.

A *RV*-object can be used to generate random samples. This will use the connected *CPD*-object to draw a sample. One can specify if the values for the conditions and parameters should be redrawn beforehand. A *CPD*-object has member functions to search for the maximum likelihood or maximum a posteriori parametrization in combination with some data set.

Once a structure of *RV*- and *CPD*-objects is created, it can be stored to disk and read again. Two files are used for this, one for describing the *RV* objects, and one for storing the *CPD*-objects.

Random number generation

A lot of the algorithms are based on pseudo-random numbers. Therefore, a good pseudo-random number generator was selected with an eye towards efficiency. We chose the Wichmann-Hill generator [74], which is based on three *linear congruential generators*, $\mathrm{lcg}(171, 30269, R^1), \mathrm{lcg}(172, 30307, R^2)$ and $\mathrm{lcg}(170, 30323, R^3)$, where $\mathrm{lcg}(a, M, R)$ generates a sequence of uniform pseudo-random numbers U_k as

$$R_{k+1} = (aR_k) \bmod M \text{ with } R_0 = R \text{ and } U_k = \frac{R_k}{M}.$$

Based on these three uniform pseudo-random numbers, a new is calculated

$$U_k = (U_k^1 + U_k^2 + U_k^3)\mathrm{mod}1.$$

This number generator has a period of almost 3.48×10^{12}, is easy to implement and gives a fast generator, even on a 16-bit machine, and showed good properties when the independence between for series of random numbers was tested experimentally.

Based on this uniform pseudo-random number generator, pseudo-random numbers with a different distribution can be generated using the inversion method (e.g., exponentially distributed numbers), the rejection method or the ratio method (e.g., standard normally distributed numbers).

Optimization and Markov Chains

An abstract multivariate function class *(MVF)* is designed. This class has member functions to compute the input-output mapping and the gradient of this mapping together with a method to update the inputs in a certain direction that makes sure that the mapping is still defined for the new input vector. This is necessary if some inputs should stay positive, like an input representing a variance parameter. The actual functions are implemented by the derived classes.

Based on this *MVF*-object, an *Optimization*-object is constructed that performs function optimization for real-valued multivariate functions using the scaled conjugate gradient algorithm [61] or a more basic gradient ascent method. Derived from this *Optimization*-object, is the *MarkovChain*-object. This implements both the Metropolis algorithm as the hybrid Markov chain Monte Carlo method for multivariate distributions. It is not necessary to normalize the distribution. The inherited *Optimization* is used to find a suitable start vector for the chain.

The classes derived from *MVF* are not indicated on Figure 8.1 because this would make the diagram less clear. Anything that needs an optimization or a Markov chain method is derived from *MVF*.

8.3.4 Bayesian networks

The *DAG* class is a structure that stores a Bayesian network, both the structure and the parameters.

The nodes of this network are represented with objects from the class *DAGNode*. Each *DAGNode* can represent either a discrete or continuous node, and has therefore both a *RVD* and a *RVC* member with an appropriate conditional probability distribution and possibly a hyper-distribution. The parents for each node are stored in an array, and also within the *RV-CPD* structure. This way, we do not have to allocate the memory for the *RV-CPD* structures each time the structure of a network is learned.

A Bayesian network can directly be used to generate a data set according to the joint distribution it represents. To perform inference, we have to convert the *DAG* first to a *CliqueTree*-object. This class is derived from the *DAG* class and can perform the inference using the *probability propagation in tree of cliques* algorithm [45].

A *DAG* can be written to disk, together with all the *RV*- and *CPD*-objects.

8.3.5 Multilayer perceptrons

The *MLP* class represents a multilayer perceptron, the multivariate input-output mapping described in Chapter 6. Both the structure and the parameters are stored, together with the prior distribution.

This class implements functions to compute the input-output mapping defined by the network and to compute the gradient of the network parameters for an input-output pattern using a certain distribution around the regression mean. In addition to this, *MLP* is derived from *MVF*. The inputs for this *MVF* are the parameters of the network, not the input of the multilayer perceptron. The output can vary from the likelihood of the model to the posterior probability using some prior, both based on the data set received from *Model*, and can be set using some attributes. This way, we have a flexible method to find the maximum likelihood or maximum a posteriori parametrizations, or sample from this a posteriori parametrization. By including the variances controlling the non-informative Gaussian prior and the regression distribution in the *MVF* inputs, we can easily optimize these parameters too.

8.3.6 Model

The last class, *Model*, contains a model in the Bayesian framework. This includes both a *DAG* and a *MLP*, which contain a prior distribution themselves, and a *DataBlock*. Together, this allows us to specify the a posteriori distribution.

Either the Bayesian network or the multilayer perceptron can be used in a stand-alone fashion, or the transformation method described in this thesis can be used. The performance of the models is computed using the area under the ROC-curve performance measure, as explained in Section 3.3.2.

8.3.7 Distributed computation

To compare the learning behaviour of the different models, we decided to generate learning curves. Such a curve displays the area under the ROC-performance when a growing number of training samples is used. The more samples that are used, the higher the performance should become, and the form of this curve tells a lot about the learning characteristics of a certain model. For each train/test proportion, we divide the data set several times, and for each division, we have to run a Markov chain to compute the Bayesian performance measure.

All this machinery consumes a considerable amount of computation time. We decided to implement these computations in a distributed manner. Fortunately, the things we have to compute can easily be divided in different jobs that can be computed by different machines. We distributed the different train/test proportions and the different data set divisions for a certain train/test proportion over different machines by using a client/server approach. The server divides the job in several subjobs. The client machines (computation machines) connect to the server and ask for a job to compute. In response, the server gives one of the subjobs to the client. Each time a client has finished his computation, it sends it back to the server, which is responsible for assembling the final result from the subjobs.

Although such packages exist, it is not such a hard task to implement, brings new experience and lots of fun, and guarantees a maximum of flexibility; both windows and Linux-based machines can be used as computation clients. We used Berkeley style TCP/IP sockets which we represented with the class *Socket*.

An abstract class *Job* was designed to exemplify a job that has to be computed. Member functions for this class include the division of a job into a series of subjobs (also *Job*-classes) and the construction of the result of the original job based on the results of the subjobs.

To send any object through a *Socket*, we constructed an abstract class *(BaseObject)* that is inherited by each class we use. This abstract class has member functions that write down an object to a bit-string or reconstruct an object from such a bit-string. This basic functionality is used for serialization and sending objects through network connections.

8.4 To conclude

A considerable amount of work was invested in the implementation of the techniques that are presented in this dissertation. This chapter gives an overview of the main components of the developed software package. Although this has no theoretical value, it gives us an overview of the used algorithms and techniques.

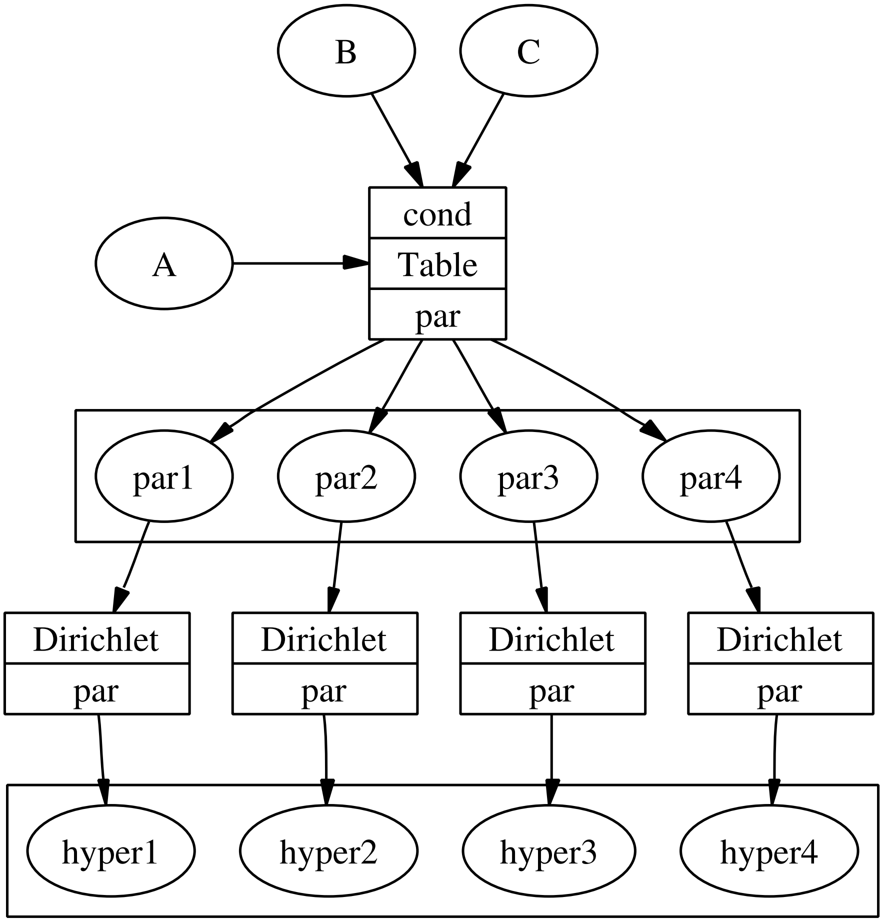


Figure 8.3: The full Bayesian representation of the dependency model indicated in Figure 8.2 with a table conditional distribution and a Dirichlet hyper-distribution. An ellipse indicates an *RV*-object, while a rectangle indicates a *CPD*-object.

Chapter 9

Conclusions and perspectives

9.1 Conclusions

The central question of this thesis was how different types of information can be combined into a model. We observed that two types of models could be distinguished; one type allows us to easily incorporate prior knowledge but has poor learning capabilities from data (donor model). Its complement also exists, a model class that learns well from numerical data but is less suitable for prior knowledge incorporation (acceptor model). A combination of both behaviours would be nice.

To meet this goal, we worked in the Bayesian framework. This framework defines probabilities as a state of knowledge about a given system. When new information is observed, this state of knowledge is updated elegantly using Bayes' rule. We designed a methodology where the prior information is collected using a donor model. Once this donor model is specified, it can be transformed to the acceptor model in the form of an a priori distribution over its parameters. Finally, this a priori distribution is updated to a posterior distribution using the statistical data set. We stated this transformation in the Bayesian framework and derived an algorithm to generate parameter vectors from this a posteriori distribution.

We chose the Bayesian network model class as donor model class and managed to incorporate prior information into this model using a few different approaches: we defined a prior distribution over the network structures from a relatedness matrix derived from expert information or textual information. We also extracted information concerning the parameters of a Bayesian network from our medical expert.

Once this donor model was specified, we transformed its information representation into an informative a priori distribution for the class of multilayer perceptrons. A technique was developed to estimate this a priori distribution, dealing appropriately with the symmetries that arise in multilayer perceptron parameter space. A whole range of distributions was introduced, ranging from mixture of Gaussians to nonlinear continuous Bayesian network models with hidden nodes.

A lot of effort was put into the development of a software package that implements *all* the necessary techniques. This includes a Bayesian network implementation with structure learning, parameter learning, and inference, a multilayer perceptron implementation with routines to perform optimization and Markov chain simulation. The transformation itself and the density estimation is also present, together with various other classes dealing with performance evaluation, distributed computations, output generation, serialization, and so on.

Finally, the described technique was applied to a real-world classification task. Pre-operative classification of an ovarian tumour as a benign or malignant tumour is of substantial importance, since it largely determines the optimal treatment that the patient should receive. By working closely together with the medical expert and participating in a rich project like the IOTA project, we found a nice application for the developed techniques. The experiments we conducted show that we are able to combine different types of information successfully, resulting in a superior classification system. The complementary properties of both the donor and the acceptor model could be combined.

9.2 Future research directions

The procedure presented in this thesis is only one way to combine different types of information. Different approaches, such as ensemble methods could and should be investigated. But our transformation technique can also be extended or refined in many directions. At first, nothing restricts us from using only Bayesian networks as donor models or multilayer perceptrons as acceptor models. Other models can be more suitable or powerful, or result in a more analytical transformation.

The special case where the donor and the acceptor model are the same seems especially promising to find a this more analytical transformation. Although this restricts the capability to combine heterogeneous information, this can still have interesting application. A major application is the combination of old and new data; numerous domains exist where similar but different data sets exist. Often, we cannot combine both data sets directly because the underlying distribution of the two data sets is different. Still, we would like to combine the information from both data sets to come to an optimal modelling.

Another keypoint we can refine, has to do with the technical details of the transformation. As such, the size of the virtual data sets needs extra investigation. It would be interesting to see the amount of information that is remained after the transformation, in function of the data set size. Also the sampling itself with the transformation technique could be optimized, using variance reduction techniques based on antithetical virtual data sets.

The different models we presented can also be subject to additional research. The main reasons we choose Bayesian networks to model the background information, apply also to other domains. The bilateral nature of a Bayesian network with a structure expressing conditional independency relationsships among the variables makes them interesting models to perform genetic network inference. In this micro-biological field, we are interested in reconstructing a biological pathway that describe the cascade of gene-activations. Through the process of activating genes based on the presence of certain proteins or external factors, a cell is able to perform numerous different functions. Understanding these pathways is a major challenge. In this micro-biological domain we have also access to heterogeneous types of information. Numerical information about the expression level of a large amount of genes can be measured using a micro-array, while numerous databases exist that contain a wealth of other information about each gene, ranging from their code to their function.

We introduced a technique to find a suitable ancestral node ordering for the Bayesian network. Although it performed well in our case, its perfromance and properties should be compared with other techniques to deal with the node order. It would be nice to know which properties a distribution has to have to trick the technique into an unwanted behaviour.

A Bayesian network is not only a promising tool to model a distribution, it can be extended into a complete modelling environment. This allows us to attach annotations to different objects of a Bayesian network, like the edges, variable values, variables or groups of variables. This extension can be helpful to organise the relevant domain information and allows us to derive valuable information about the structure of the domain [6].

On the multilayer perceptron side we can also proceed. This thesis presented a method to estimate distributions in neural network weightspace and proposed a range of increasingly more flexible densities. The effect of this increasing flexibility can be investigated further, especially the usage of the Bayesian network based density with hidden nodes. Although the symmetry elimination procedure we suggested, based on the EM algorithm and our heuristic, performed well for the experiments we conducted, their behaviour should be tested on larger networks.

From the application point of view, we developed useful models for the ovarian tumour problem. Currently, other techniques are being tested [59], and they should be compared with each other in a valid manner. Our models can be refined by using other variable discretization techniques, dealing with missing values or trying to perform a more detailed classification by predicting the subclass of a tumour, instead of classifying tumours as benign or malignant.

In addition to this, the IOTA project is not finished and exciting new information is underway; the IOTA project collected several tumours, which are now analyzed using micro-array technology. This technology gives us information about the genes that are expressed in the tumour. How to use this extra information to build a deeper understanding of the problem poses many questions and is certainly interesting.

Finally, the software can be a last topic where this research can be continued.

It forms a nice starting place to develop similar applications, extending it to large-scale problems, other models, or build a graphical user interface for it. This could eventually lead to medical doctors using the described models to assist them in triaging patients pre-operatively.

Appendix A

Variable descriptions

Age

A discrete-valued, but continuous variable indicating the age of the patient, measured in years.

Discretization intervals:

Value	Interval
< 40	$[0, 40[$
40-50	$[40, 50[$
50-60	$[50, 60[$
60-70	$[60, 70[$
$70 \leq$	$[70, 120[$

Ascites

A binary variable indicating the presence of ascites — fluid outside the pouch of Douglas. The amount of fluid is indicated with the variable *Fluid*.

Bilateral

Binary variable that indicates if the patient has a bilateral tumour, a tumour on both ovaries. Only the dominant tumour is kept in our data set.

CA125

The measurement of the tumour marker CA125, using the CA125 II immunoradiometric assay. The measurements are in U/ml. This variable is continuous and its logit plot (Figure A.1) indicates a logarithmic transformation before using this variable in logistic regression or multilayer perceptron models. Figure A.2 show this logit plot after the logarithmic transformation. Each odds on this logit plot has been estimated using one tenth of the data samples. The mean of this bin is indicated on the x axis, while the y axis indicates the logarithm of the estimated odds.

Discretization intervals:

Value	Interval
< 35	$[1, 35[$
35-65	$[35, 65[$
$65 \leq$	$[65, 1\,000\,000[$

Colour Score

A subjective semiquantitative assessment of the amount of blood flow (area and colour scale) within the septa, cyst walls, or solid tumour areas.

The following four gradations are used:

1. No blood flow can be found in the lesion.
2. Only minimal flow can be detected.

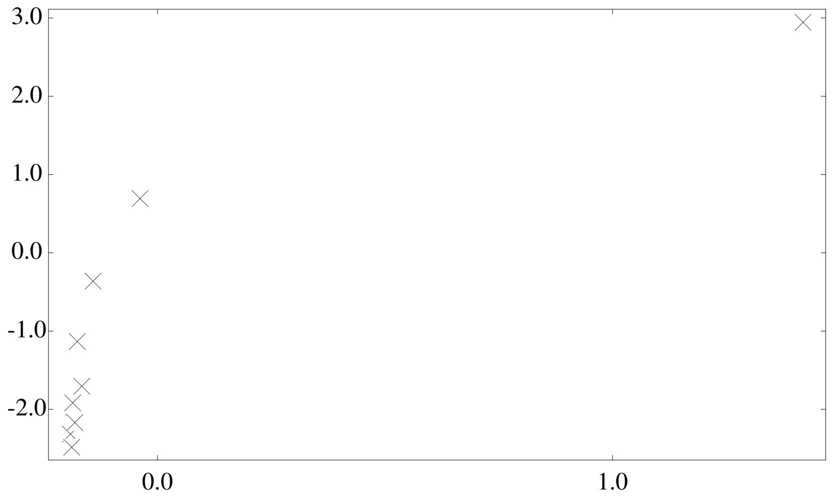


Figure A.1: The logit plot for the original *CA125* variable after a linear transformation to normalize the variable to unit variance. This plot suggests a logarithmic transformation.

3. Moderate flow is present.
4. Adnexal mass appears highly vascular with marked blood flow using colour Doppler. This colour score refers only to colour Doppler image and not to Doppler shift spectrum.

The blood flow is further characterized with the variables *PI*, *RI*, *PSV* and *TAMXV*. This variable is discrete but ordered.

Fluid

The amount of fluid in the pouch of Douglas, measured in a sagittal plane. The largest anteroposterior diameter is given in millimetres. This variable is continuous in nature. Figure A.3 shows the logit plot for this variable. Each bin contains the same amount of data records to estimate the necessary odds, except for the left most bin. This bin corresponds to those observations with no fluid measured, which contains 560 records.

The discretized version of this variable only indicates if fluid was found:

Value	Interval
No	$[0, 1[$
Yes	$[1, 1000[$

Hormone therapy

A binary variable that indicates if a hormone therapy has been applied.

Incomplete septum

An incomplete septum is defined as a thin strand of tissue running across the cyst cavity from one internal surface to the contralateral side, but which is not complete in some scanning planes. The presence of an incomplete septum is represented with this binary variable. If the septum is complete, its thickness is indicated with the variable *Septum*.

Locularity

A nominal variable describing the morphology of the tumour using five categories:

- *Unilocular*: a unilocular cyst without septa and without solid parts or papillary structures
- *Unilocular with solid component*: unilocular cyst with a measurable solid component or at least one papillary structure

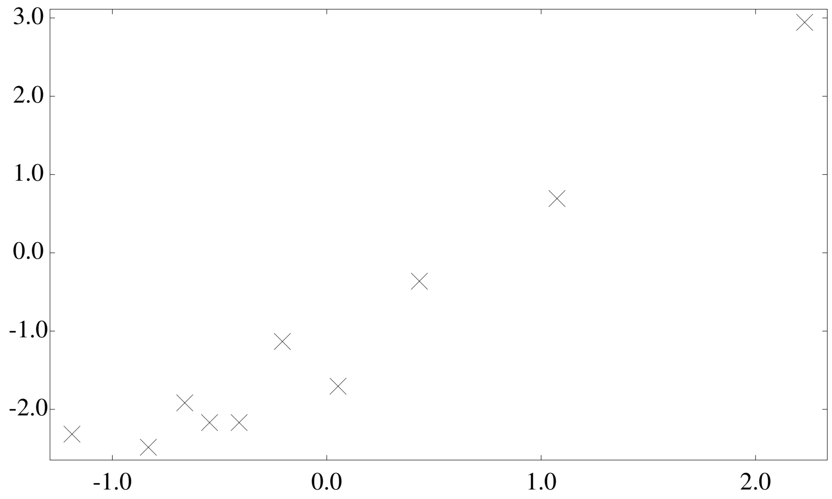


Figure A.2: The logit plot for the *CA125* variable, *after* a logarithmic transformation. This plot shows a linear relation between the *CA125* variable and the logarithm of the odds. *CA125* has been normalized to unit variance *after* the logarithmic transformation.

- *Multilocular*: a cyst with at least one septum but no measurable solid components or papillary projections
- *Multilocular with solid component*: a multilocular cyst with a measurable solid component or at least one papillary structure
- *Solid*: a tumour where the solid components comprise 80% or more of the tumour when assessed in a two-dimensional section

Menopausal status

A three-valued nominal variable describing the menopausal status. The possible values are:

- *Pre-menopause*: Before the period of natural cessation of menstruation
- *Hysterectomy*: This value indicates that the uterus has been surgically removed
- *Post-menopause*: After the menopause

Pulsatility index (PI)

One of the parameters describing the blood flow characteristics of the tumour, together with *RI*, *PSV*, and *TAMXV*. This parameter describes the pulsation of the blood in the tumour and is recorded with colour Doppler imaging technology. This variable is only valid if there is actual blood flow present (*Colour Score* $\neq 1$).

Discretization intervals:

Value	Interval
No blood flow	
Low	$[0, 1[$
High	$[1, 1000[$

Peak systolic velocity (PSV)

Another parameters describing the blood flow characteristics of the tumour, together with *PI*, *RI*, and *TAMXV*. This parameter describes the peak systolic velocity of the blood in the tumour, measured in centimetres per second, and is recorded with colour Doppler imaging technology. This variable is only valid if there is actual blood flow present (*Colour Score* $\neq 1$).

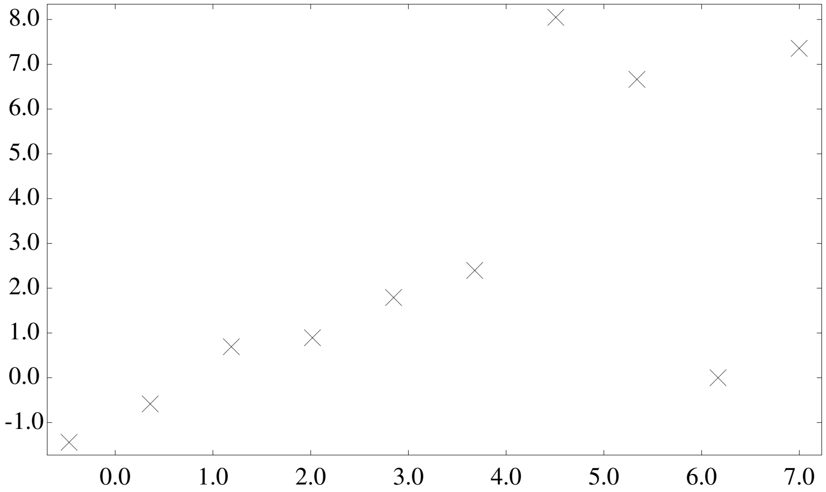


Figure A.3: The logit plot for the variable *Fluid* after a linear transformation to normalize the variable to unit variance. This plot indicates a linear relation with the odds of the output variable.

The logit plot Figure A.4 suggests a logarithmic transformation. Figure A.5 show the logit plot after this transformation.

Discretization intervals:

Value	Interval
No blood flow	
Low	$[0, 20[$
High	$[20, 1000[$

Pain

A binary variable indicating the presence of pelvic pain during the scan.

Papillation

A binary variable indicating the presence of papillary projections into the cyst cavity.

Papillation flow

This binary variable indicates if flow is present within any of the papillary projections. This variable is only valid if there are papillary projections present, as indicated by the variable *Papillation*.

Papillation smooth

This binary variable describes the shape of the papillary projections. They can be *smooth* or *irregular*. This variable is only valid if there are papillary projections present, as indicated by the variable *Papillation*.

Parity

A discrete variable indicating the number of deliveries. No difference is made between three or more deliveries, resulting in four discretization bins.

Pathology

The binary variable indicating if the tumour was found to be benign or malignant. This is the variable of interest that provides the class label and that we try to predict. Tumours are classified according to the criteria recommended by the International Federation of Gynecology and Obstetrics [69].

Personal history breast cancer

A binary variable indicating the patient's personal history with respect to breast cancer.

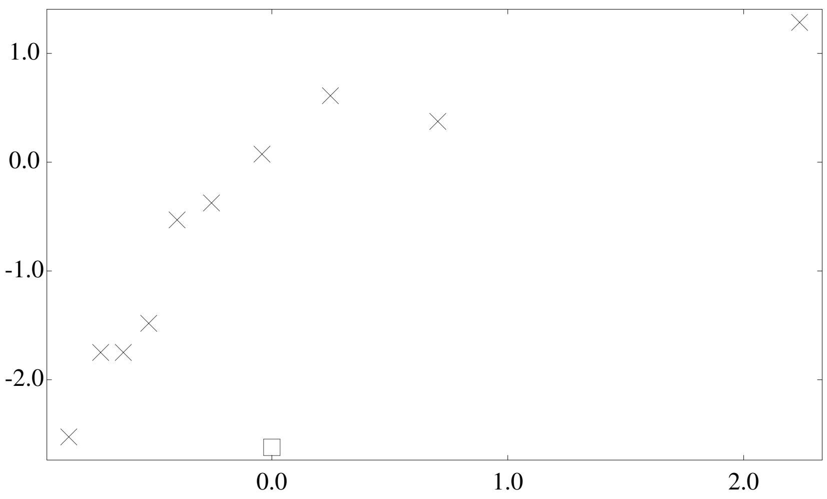


Figure A.4: The logit plot for the *PSV* variable, indicating a logarithmic transformation. On the x axis, the mean of each bin is indicated, while the y axis represents the logarithm of the odds estimated for the records within each bin. Each bin contains the same number of data records. The bin indicated with the square contains those tumours where no blood flow was present, resulting in an irrelevant *PSV* variable.

Pill Use

The total years of contraceptive use of the pill, measured in years. In the collection protocol, this was an optional variable, which unfortunately resulted in a lot of missing values.

Discretization intervals:

Value	Interval
< halfyear	$[0.0, 0.5[$
0.5-5 year	$[0.5, 5.1[$
5 years <	$[5.1, 100[$

Resistance index (RI)

One of the parameters describing the blood flow characteristics of the tumour, together with *PI*, *PSV*, and *TAMXV*. This parameter describes the resistance index of the blood in the tumour and is recorded with colour Doppler imaging technology. This variable is only valid if there is actual blood flow present ($\textit{Colour Score} \neq 1$).

Discretization intervals:

Value	Interval
No blood flow	
Low	$[0, 0.5[$
High	$[0.5, 1000[$

Septum

A septum is defined as a thin strand of tissue running across the cyst cavity from one internal surface to the contralateral side. The thickness of the thickest septum is measured in millimetre where it appears to be at its widest. The logit plot for this variable (Figure A.6), reveals two things. At first, it shows that the tumours where *Septum* is 0 do not follow the general behaviour of the other tumours. Therefore, a special design variable was introduced for this variable to deal with the case $\textit{Septum} = 0$. Second, the logit plot suggests a logarithmic

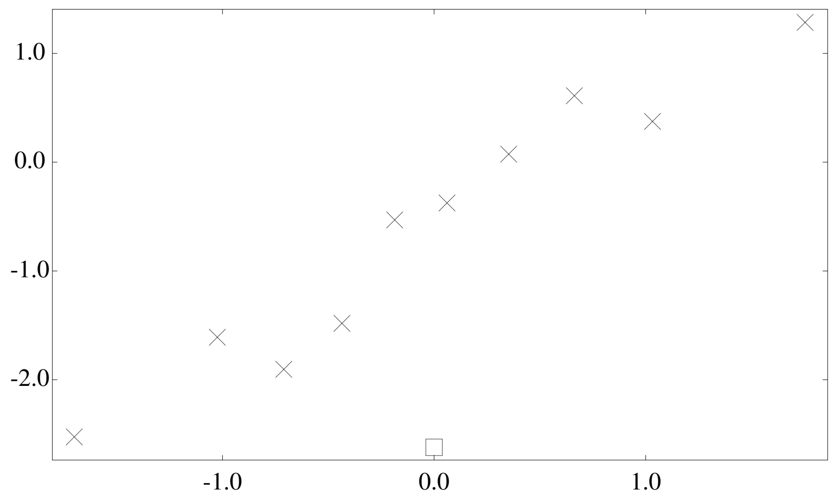


Figure A.5: The logit plot for the *PSV* variable, *after* the logarithmic transformation. On the x axis, the mean of each bin is indicated, while the y axis represents the logarithm of the odds estimated for the records within each bin. Each bin contains the same number of data records. The bin indicated with the square contains those tumours where no blood flow was present, resulting in an irrelevant *PSV* variable.

transformation. Figure A.7 shows the logit plot after this transformation. Note that the case $Septum = 0$ is already treated with a design variable.

This variable is discretized by remaining only the presence or absence of a septum.

Shadows

This binary variable indicates the presence of acoustic shadows, defined as loss of acoustic echo behind a sound-absorbing structure.

Solid

The binary variable *Solid* means echogenicity suggesting the presence of tissue. Methods to distinguish between blood clots and the presence of solid tissue are the use of colour Doppler and to look for internal movement when gently pushing to the structure with the transducer.

Time averaged maximum velocity (TAMXV)

One of the parameters describing the blood flow characteristics of the tumour, together with *PI*, *RI*, and *PSV*. This parameter describes the time averaged maximum velocity of the blood in the tumour and is recorded with colour Doppler imaging technology. This variable is only valid if there is actual blood flow present ($Colour\ Score \neq 1$). The logit plot for this variable (Figure A.8) indicates a logarithmic transformation. The logit plot after this transformation is shown in Figure A.9. The square indicates the group of tumours where no blood flow is present, leaving this variable irrelevant.

Discretization intervals:

Value	Interval
No blood flow	
Low	$[0, 15[$
High	$[15, 1000[$

Volume

This continuous variable describes the volume of the tumour. The logit plot for this variable indicates a logarithmic transformation (see Figure A.10).

Discretization intervals:

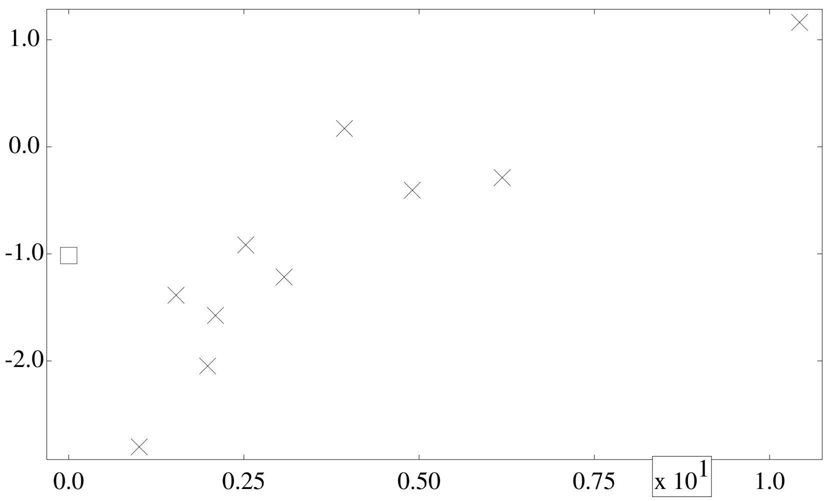


Figure A.6: The logit plot for the *Septum* variable, indicating a design variable for the case ($Septum = 0$) and a logarithmic transformation. On the x axis, the mean of each bin is indicated, while the y axis represents the logarithm of the odds estimated for the records within each bin. Each bin contains the same number of data records. The bin indicated with the square contains those tumours where no septum was present.

Value	Interval
< 50	$[0, 50[$
50-400	$[50, 400[$
$400 \leq$	$[400, 100000[$

Wall regularity

A three-valued nominal variable describing the structure of the internal wall of the cyst. This wall can be *regular*, *irregular* or *not visible*.

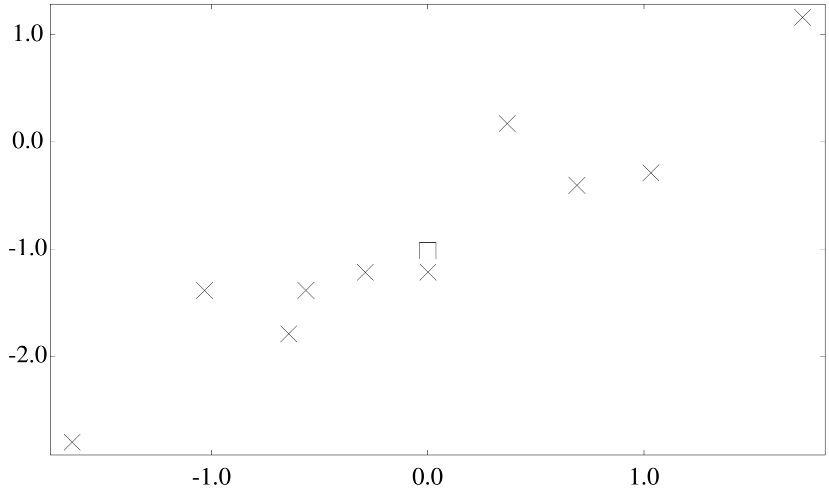


Figure A.7: The logit plot for the *Septum* variable, *after* the introduction of a design variable dealing with the special case (*Septum* = 0) and *after* a logarithmic transformation. On the x axis, the mean of each bin is indicated, while the y axis represents the logarithm of the odds estimated for the records within each bin. Each bin contains the same number of data records. The bin indicated with the square contains those tumours where no septum was present.

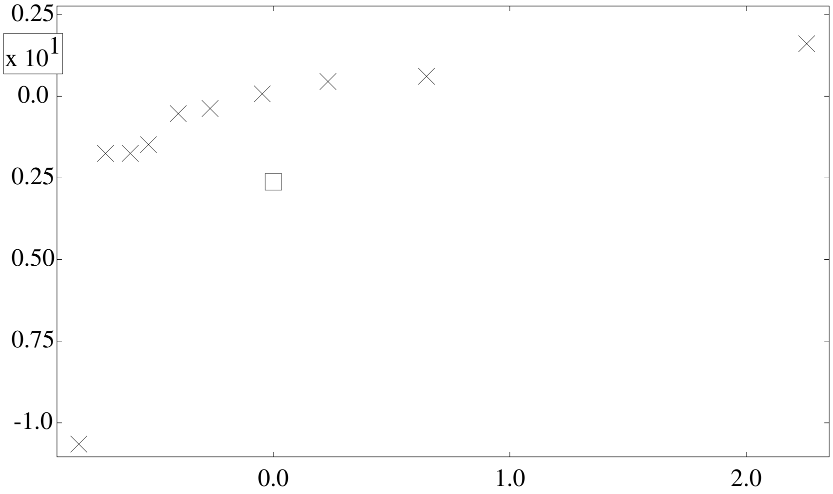


Figure A.8: The logit plot for the *TAMXV* variable, indicating a logarithmic transformation. On the x axis, the mean of each bin is indicated, while the y axis represents the logarithm of the odds estimated for the records within each bin. Each bin contains the same number of data records. The bin indicated with the square contains those tumours where no blood flow was present, resulting in an irrelevant *TAMXV* variable.

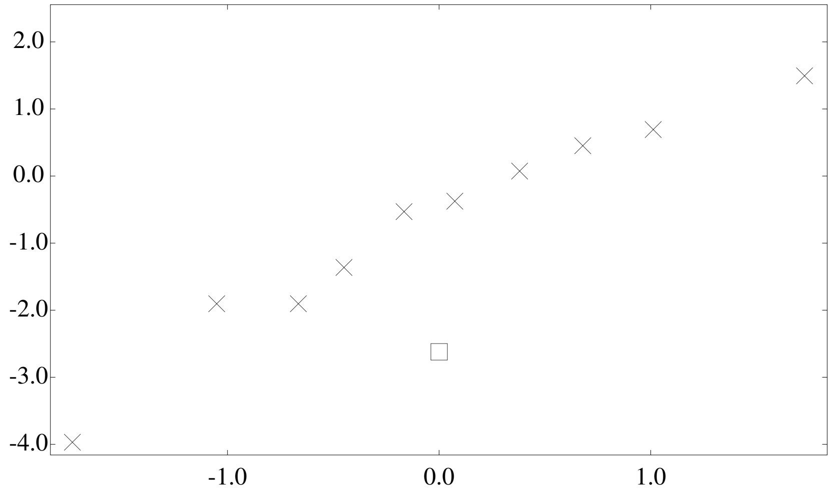


Figure A.9: The logit plot for the *TAMXV* variable, *after* the logarithmic transformation. On the x axis, the mean of each bin is indicated, while the y axis represents the logarithm of the odds estimated for the records within each bin. Each bin contains the same number of data records. The bin indicated with the square contains those tumours where no blood flow was present, resulting in an irrelevant *TAMXV* variable.

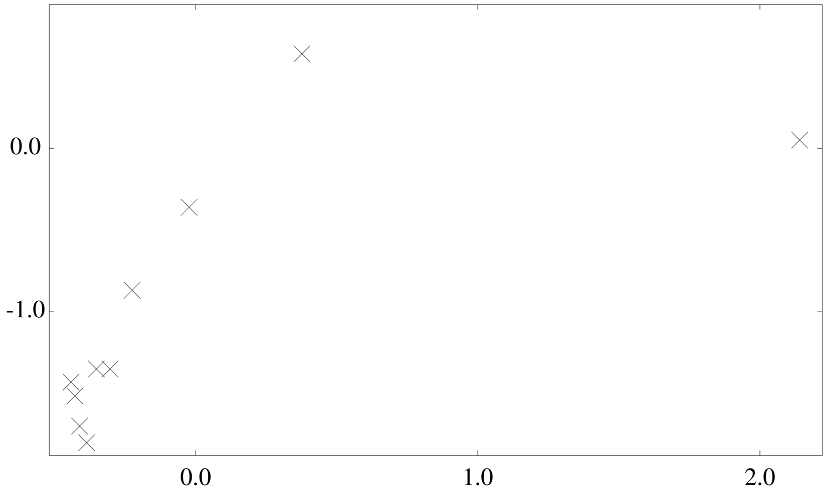


Figure A.10: The logit plot for the *Volume* variable. On the x axis, the mean of each bin is indicated, while the y axis represents the logarithm of the odds estimated for the records within each bin. Each bin contains the same number of data records.

Bibliography

[1] Y. S. Abu-Mostafa. Hints and the VC dimension. *Neural Computation*, 5(2):278–288, 1993.

[2] L. Ameraal. *C++*. Academic service, 1993.

[3] P. Antal, G. Fannes, Y. Moreau, B. De Moor, J Vandewalle, and D. Timmerman. Extended Bayesian regression models: a symbiotic application of belief networks and multilayer perceptrons for the classification of ovarian tumors. In *Lecture Notes in Artificial Intelligence (AIME 2001)*, pages 177–187, 2001. Cascais, Portugal.

[4] P. Antal, G. Fannes, F. De Smet, and B. De Moor. Ovarian cancer classification with rejection by Bayesian belief networks. In *Workshop notes on Bayesian Models in Medicine, European Conference on Artificial Intelligence in Medicine (AIME'01)*, pages 23–27, 2001. Cascais, Portugal.

[5] P. Antal, G. Fannes, H. Verrelst, B. De Moor, and J. Vandewalle. Incorporation of prior knowledge in black-box models : Comparison of transformation methods from bayesian network to multilayer perceptrons. In *Proceedings of the Workshop on Fusion of Domain Knowledge with Data for Decision Support, The Sixteenth Conference on Uncertainty in Artificial Intelligence, Stanford University, USA*, pages 7–12, 2000.

[6] P. Antal, T. Meszaros, B. De Moor, and T. Dobrowiecki. Annotated bayesian networks : A tool to integrate textual and probabilistic medical knowledge. In *Proceedings of the Fourteenth IEEE Symposium on Computer-Based Medical Ssytems (CBMS 2001)*, pages 177–182, Bethesda, MD, july 2001.

[7] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[8] Rev. T. Bayes. An essay toward solving a problem in the doctrine of chances. *Philosopical Transactions of London*, 53:370–418, 1763.

[9] J. M. Bernardo. *Bayesian Theory*. Wiley & Sons, 1995.

[10] J. A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report tr-97-021, International Computer Science Institute, Berkeley, California, USA, 1997.

[11] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

[12] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, October 1989.

[13] G. E. P. Box. Non-normality and tests on variance. *Biometrika*, 40:318–335, 1953.

[14] A. Burgun and O. Bodenreider. Methods for exploring the semantics of the relationships between co-occurring umls concepts. In *Proceedings of the World Congress on Health and Medical Informatics (MedInfo 2001)*, pages 171–175, 2001.

[15] E. Castillo, J. M. Gutiérrez, and A. S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer, New York, first edition, 1997.

[16] M. H. Chen, Q. M. Shao, and J. G. Ibrahim. *Monte Carlo methods in Bayesian computations.* Springer series in statistics, 2000.

[17] D. M. Chickering. A transformational characterization of equivalent Bayesian network structures. In Besnard, Philippe and Steve Hanks, editors, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI'95)*, pages 87–98, San Francisco, CA, USA, August 1995. Morgan Kaufmann Publishers.

[18] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Journal of Artificial Intelligence*, 42:393–405, 1990.

[19] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Journal of Machine Learning*, 9:309–347, 1992.

[20] R. T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14(1):1–13, 1946.

[21] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. M. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the world wide web. *Journal of Artificial Intelligence*, 118(1-2):69–113, 2000.

[22] S. Dasgupta. The sample complexity of learning fixed-structure Bayesian networks. *Journal of Machine Learning*, 29:165–180, 1997.

[23] M. H. DeGroot. *Optimal Statistical Decisions.* McGraw-Hill, New York, 1970.

[24] H. M. Deitel and P. J. Deitel. *How to program C++.* Prentice Hall, 2003.

[25] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer, New York, 1996.

[26] W. B. Frakes. *Information retrieval: Data Structures and Algorithms*, chapter Stemming Algorithms. Prentice Hall, 1992.

[27] B. J. Frey and G. E. Hinton. Variational learning in nonlinear Gaussian belief networks. *Journal of Neural Computation*, 11(1):193–213, 1999.

[28] N. Friedman. The Bayesian structural EM algorithm. In Gregory F. Cooper and Serafín Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 129–138, San Francisco, July 24–26 1998. Morgan Kaufmann.

[29] N. Friedman, M. Goldszmidt, D. Heckerman, and S. Russel. Challenge: Where is the impact of Bayesian networks in learning. In *Proc. Fifteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 1997.

[30] N. Friedman and D. Koller. Being bayesian about network structure. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 201–210, SF, CA, June 30– July 3 2000. Morgan Kaufmann Publishers.

[31] N. Friedman and Y. Singer. Efficient Bayesian parameter estimation in large discrete domains. In *Proceedings of Neural Information Processing Systems (NIPS 98)*, 1998.

[32] N. Friedman and Z. Yakhini. On the sample complexity of learning Bayesian networks. In *Proceedings of the 12th Uncertainty in Artificial Intelligence Conference (UAI96), Portland, Oregon, USA*, pages 274–282, 1996.

[33] D. Geiger and D. Heckerman. Learning bayesian networks. Technical Report MSR-TR-95-02, Microsoft Research (MSR), December 1994.

[34] D. Geiger and D. Heckerman. Learning gaussian networks. Technical Report MSR-TR-94-10, Microsoft Research (MSR), July 1994.

[35] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis.* Chapman & Hall, 1995.

[36] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[37] J. M. Hammersley and D. C. Handscomb. *MonteCarlo Methods.* Methuen & Co Ltd, 1964.

[38] D. J. Hand. *Construction and assessment of classification rules.* Wiley series in probability and statistics, 1997.

[39] D. Haussler. Quantifying inductive bias: AI learning algorithms and valiant's learning framework. *Artificial Intelligence*, 36(2):177–221, September 1988.

[40] D. Haussler, M. Kearns, and R. Schapire. Bounds on the sample complexity of Bayesian learning using information thjeory and the VC dimension. *Machine Learning*, 14(1):83–113, 1994.

[41] D. Heckerman. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research (MSR), March 1995.

[42] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Journal of Neural Networks*, 4(2):251–257, April 1991.

[43] D. W. Hosmer and S. Lemeshow. *Applied Logistic Regression.* John Wiley & Sons, New York, 1989.

[44] R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, pages 721–762. Strategic Decisions Group, Menlo Park, California, 1984.

[45] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

[46] I. Jacobs. A risk of malignancy index incorporating CA125, ultrasound and menopausal status for the acurate preoperative diagnosis of ovarian cancer. *Obstetrics and Gynaecology*, 97:922–929, 1990.

[47] E. T. Jaynes and G. L. Bretthorst (editor). *Probability theory, the logic of science.* Cambridge Press, 2003.

[48] H. Jeffreys. *Probability Theory.* Clarendon Press, Oxford, 1939.

[49] T. Jenssen, L. M. J. Oberg, M. L. Anderson, and J. Komorowski. Methods for large-scale mining of networks of human genes. In *Proceedings of the SIAM International Conference on Data Mining*, page 16, 2001.

[50] T. K. Jenssen, A. Læreid, J. Komorowski, and E. Hovig. A literature network of human genes for high-throughput analysis of gene expression. *Nature Genetics*, 28(1):21–28, May 2001.

[51] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Journal of Machine Learning*, 37(2):183–233, 1999.

[52] M. J. Jordan. *Learning in graphical models.* Kluwer, 1999.

[53] T. Kohonen. Self-organised formation of topological correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[54] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78:1464–1480, 1990.

[55] D. Koller, U. Lerner, and D. Angelov. A general algorithm for approximate inference and its application to hybrid bayes nets. In Kathryn B. Laskey and Henri Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 324–333, S.F., Cal., July 30–August 1 1999. Morgan Kaufmann Publishers.

[56] A. N. Kolmogorov. *Foundations of the theory of probability.* Chelsea Publishing Co., New York, 1950.

[57] P. S. Laplace. *Théorie analytique des probabilités.* Courcier Imprimeur, Paris, 1812.

[58] S. L. Lauritzen and D. J. Spigelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. R. Statist. Soc., Series B, Vol. 50, :2, 157-224*, 1988.

[59] C. Lu, T. Van Gestel, J. Suykens, S. Van Huffel, I. Vergote, and D. Timmerman. Classification of ovarian tumor using Bayesian least squares support vector machines. In M. Dojat E. Keravnou and P. Barahona, editors, *Procceedings of the 9th Conference on Artificial Intelligence in Medicine in Europe (AIME 2003)*, volume 2780, pages 219–228. Springer-Verlag, oct 2003.

[60] D. J. C. MacKay. Probable networks and plausible predictions - a review of practical bayesian methods for supervised neural networks. *Network*, 6:469–505, 1995?

[61] M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Journal of Neural Networks*, 6:525–533, 1993.

[62] S. Monti and G. F. Cooper. Learning bayesian belief networks with neural network estimators. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 578. The MIT Press, 1997.

[63] R. M. Neal. *Bayesian Learning for Neural Networks*. Number 118 in Lecture Notes in Statistics. Springer, New York, USA, 1996.

[64] R. M. Neal. Transferring prior information between models using imaginary data. Technical Report tr-0108, Dept. of Statistics, University of Toronto, 2001.

[65] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants, 1993.

[66] P. Niyogi, F. Girosi, and T. Poggio. Incorporating prior information in machine learning by creating virt ual examples, 1998.

[67] J. C. Park. Using combinatory categorical grammar to extract biomedical information. *IEEE Intelligent Systems*, 16(6):62–67, 2001.

[68] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[69] F. Petersson. Annual report on the results of treatment in gynecological cancer. Twenty-first volume statements of results obtained in patients 1982 to 1986, inclusive 3- and 5-year survival up to 1990. *International Journal of Gynaecology and Obstetrics*, pages S238–S277, 1991.

[70] D. Proux, F. Rechenmann, and L. Julliard. A pragmatic information extraction strategy for gathering data on genetic interactions. In Russ Altman, L. Bailey, Timothy, Philip Bourne, Michael Gribskov, Thomas Lengauer, and Ilya N. Shindyalov, editors, *Proceedings of the 8th International Conference on Intelligent Systems for Molecular (ISMB-00)*, pages 279–285, Menlo Park, CA, August 16–23 2000. AAAI Press.

[71] S. Ramachandran. Theory refinement of bayesian networks with hidden variables. Technical Report AI98-265, The University of Texas at Austin, Department of Computer Sciences, May 1 1998.

[72] F. L. Ramsey and D. W. Schafer. *The Statistical Sleuth, A Course in Methods of Data Analysis*. Duxbury, 1997.

[73] T. C. Rindflesch, L. Tanabe, and J. N. Weinstein. Edgar: Extraction of drugs, genes and relations from the biomedical literature. In *Proceedings of the Pacific Symposium on Biocomputing (PSB00)*, volume 5, pages 514–525, 2000.

[74] B. D. Ripley. *Stochastic Simulation*. Wiley, New York, 1983.

[75] S. M. Rüger and A. Ossen. Clustering in weight space of feedforward nets. *Lecture Notes in Computer Science*, 1112:83–88, 1996.

[76] S. J. Russell, J. Binder, D. Koller, and K. Kanazawa. Local learning in probabilistic networks with hidden variables. In *IJCAI*, pages 1146–1152, 1995.

[77] J. Sjoberg and L. Ljung. Overtraining, regularization and searching for minimum with applications to neural networks. *International Journal of Control*, 62:1391–1407, 1995.

[78] D. J. Spiegelhalter and R. G. Cowell. Learning in probabilistic expert systems. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics*, volume 4, pages 447–465. Oxford University Press, 1992.

[79] B. Stroustrup. *The C++ programming language.* Addison-Wesley, 2000.

[80] D. R. Swanson and N. R. Smalheiser. An interactive system for finding complementary literatures: a stimulus to scientific discovery. *Journal of Artificial Intelligence*, 91:183–203, 1997.

[81] D. Timmerman. *Ultrasonography in the assessment of ovarian and tamoxifen-associated endometrial pathology.* Ph.D. dissertation, Leuven University Press, D/1997/1869/70, 1997.

[82] D. Timmerman. Artificial neural network models for the pre-operative discrimination between malignant and benign adnexal masses. . *Ultrasound in Obstetrics and Gynecology*, 13:17–25, 1999.

[83] D. Timmerman, L. Valentin, T. H. Bourne, W. P. Collins, H. Verrelst, and I. Vergote. Terms, definitions and measurements to describe the sonographic features of adnexal tumors: a consensus opinion from the international ovarian tumor analysis (IOTA) group. *Ultrasound in Obstetrics and Gynecology*, 16(5):500–505, oct 2000.

[84] S. Tingulstad, B. Hagen, F. E. Skjeldestad, M. Onsrud, T. Kiserud, T. Halvorsen, and K. Nustad. Evaluation of a risk of malignancy index based on serum CA125, ultrasound findings and menopausal status in the pre-operative diagnosis of pelvic masses. *Obstetrics and Gynaecology*, 103:826–831, 1996.

[85] G. G. Towell and J. W. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1–2):119–165, 1994.

[86] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[87] J. Venn. *The logic of chance.* MacMillan & Co., London, 1866.

[88] W. Wiegerinck. Variational approximations between mean field theory and the junction tree algorithm. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 626–633, SF, CA, June 30– July 3 2000. Morgan Kaufmann Publishers.

Curriculum Vitae

Personal information

Name: Geert Fannes
Address: Maarschalk Fochplein 9 bus 5, B-3000 Leuven
Born: Nice (Fr.), 05/01/1978
Nationality: Belgian
Gender: Male
Married to: Annemie Janssens
Telephone number: 016/29 64 74
GSM: 0498/76 27 08
Email: geert.fannes@esat.kuleuven.ac.be

Education

High school: 1989-1995

Division Mathematics-Science (8 hours mathematics)
Sint-Martinus College, Waversesteenweg 96, B-3090 Overijse

Limburgs Universitair Centrum (LUC): 1995 – 1997

09/06/96 1ste kan Wiskunde, grootste onderscheiding

12/07/97 2de kan Wiskunde, grote onderscheiding

Katholieke Universiteit Leuven (KUL): 1997 – 1999

03/07/98 1ste lic Wiskunde, grote onderscheiding

08/07/99 2de lic Wiskunde, grote onderscheiding

Research

Aspirant FWO: 1999 – 2003

Research was performed at the Department of Electronical Engineering (ESAT/SCD), Kasteelpark Arenberg 10, B-3001 Heverlee, Katholieke Universiteit Leuven, supervised by Prof. Dr. Ir. Bart De Moor and Prof. Dr. Ir. Joos Vandewalle.
Title: *Bayesian learning with expert knowledge: Transforming informative priors between Bayesian networks and multilayer perceptrons*