KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science,
Engineering & Technology
Faculty of Engineering
Department of Electrical Engineering

Kernel based methods for microarray and mass spectrometry data analysis

Fabian OJEDA

Dissertation presented in partial
fulfillment of the requirements for
the degree of Doctor
in Engineering

May 2011

Kernel based methods for microarray and mass spectrometry data analysis

Fabian OJEDA

Jury:
Prof. dr. ir. P. Sas, chair
Prof. dr. ir. B. De Moor, promotor
Prof. dr. ir. J.A.K. Suykens, co-promotor
Prof. dr. ir. Y. Moreau
Prof. dr. J. Rozenski
Prof. dr. M. Van Barel
Prof. dr. ir. G. Bontempi
(Université Libre de Bruxelles)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering

May 2011

D/2011/7515/59
ISBN 978-94-6018-357-7

A mi madre, padre y hermanos.

Abstract

This thesis studies the integration of a particular family of machine learning methods into the analysis of high dimensional data sets from microarray and mass spectrometry technologies. More precisely, the class of regularized least squares methods and kernel-based machine learning methods are considered. Kernel-based methods integrate techniques from convex optimization, functional analysis and statistical learning theory into a powerful yet simple framework, and have enjoyed in the last decade great success over a wide range of application tasks in bioinformatics, financial engineering, text mining, image processing and time series prediction.

At the beginning of this thesis fundamentals and principles on regularized learning and kernel-based methods are reviewed. After this, we concentrate on support vector machines (SVM) methods and, particularly, the Least-Squares Support Vector Machines (LS-SVM) formulations. Based on the structure of the LS-SVM classifiers, we propose a novel algorithm for variable selection that makes use of low rank matrices to update the LS-SVM parameters. This method provides efficient simplifications for linear kernels that can scale up to thousands of variables and it is therefore applicable in microarray analysis. Subsequently, we provide extensions of this framework to polynomial kernels using additive structures. Existing methods discard the use of resampling techniques due to its computational complexity. We provide, however, efficient implementations of the leave-one-out (LOO) estimator which is in turn the mechanism to assess the relevance of the variables.

Later on we turn our point of interest on the application of regularized learning methods to Mass Spectral Imaging (MSI) data. We present an approach to learn sparse predictive models that integrate structural information from MSI data. The goal is to develop (semi)-supervised models that use the labeled portions

of the tissue to help predict the anatomical labels for the unlabeled regions. The medical objective of such models would be, for instance, to provide the pathologist with insight in interpreting molecular tissue content of areas that do not lend themselves for straightforward human classification. Particularly we address issues regarding inherent ordering of the model variables and the spatial relationships of the data points. Furthermore, to overcome the lack of labeled data points we exploit the prior assumption that nearby spectra are likely to have the same label. Numerically, the proposed model is shown to be equivalent to a LASSO formulation and therefore can be efficiently solved via the LARS (Least Angle Regression) algorithm. Moreover, each component in our optimization problem clearly embodies the structural information of MSI data.

As a last application of kernel methods in this thesis, we deal with the problem of clustering genes from microarray data. To this end, we explore the use of a spectral clustering formulation that incorporates an extension for out-of-sample points. In our approach, an informative small subset genes is first selected via entropy maximization, and subsequently the clustering model is used to infer cluster memberships for the remaining genes.

Abbreviations

ARD	Automatic Relevance Determination
ARI	Adjusted Rand Index
AUC	Area Under the Curve
CH	Caliński and Harabasz index
CV	Cross-Validation
ENET	Elastic Net algorithm
FDA	Fisher Discriminant Analysis
GRN	Gene Regulatory Network
KKT	Karush-Kuhn-Tucker
LARS	Least Angle RegreSsion
LASSO	Least Absolute Shrinkage Selection Operator
LOO	Leave-One-Out
LRLSVM	Low Ranked LS-SVM
LS-SVM	Least Squares Support Vector Machine
MS	Mass Spectrometry
MSE	Mean Square Error
MSI	Mass Spectral Imaging
Ncut	Normalized Cut

OLS	Ordinary Least Squares
OrLS	Orthogonal Least Squares
PCA	Principal Component Analysis
PRESS	Predicted Residual Sum of Squares
QP	Quadratic Programming
RBF	Radial Basis Function
RFE	Recursive Feature Elimination
RLS	Regularized Least Squares
ROC	Receiver Operating Characteristic curve
SBE	Sequential Backward Elimination
SFS	Sequential Forward Selection
SMW	Sherman-Morrison-Woodbury formula
SVM	Support Vector Machine
TF	Transcription Factor

List of Symbols

$\operatorname{argmin}_x f(x)$ Minimization over x, optimal value of x returned

$\mathbf{1}_n$ n-dimensional vector with entries equal to 1

$\mathbf{I}_n$ $n \times n$ identity matrix

$\mathbf{L}$ Lower triangular Cholesky factor

$\mathbf{w} = [w_1, \ldots, w_d]^\top \in \mathbb{R}^d$ Weight vector

$\mathbf{x}^k = [x_1^k, x_2^k, \ldots, x_n^k]^\top$ Describes the variable k

$\mathbf{x}^\top$ Transpose of the vector $\mathbf{x}$

$\boldsymbol{\mathcal{L}}$ Laplacian matrix

$\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^d$ Training data set of n input-output data points

$\mathcal{F}$ Feature space of dimension d_h

$\mathcal{S}$ Set of objects (e.g. set of vectors, indices)

$\mathcal{X}$ Input space of dimension d

$\min_x f(x)$ Minimization over x, minimal function value returned

$\operatorname{diag}(\mathbf{X})$ Diagonal of matrix $\mathbf{X}$

$\varphi(\cdot)$ Feature map

$\lvert\mathcal{S}\rvert$	Number of elements in set $\mathcal{S}$
$\mathbf{\Omega}^\top$	Transpose of the matrix $\mathbf{\Omega}$
$\mathbf{\Omega}_{ij}$	ij-th entry of the matrix $\mathbf{\Omega}$
$K(\mathbf{x}_i, \mathbf{x}_j)$	Mercer kernel evaluated between data points $\mathbf{x}_i, \mathbf{x}_j$
m/z	mass-to-charge ratio
$x_i^k \in \mathbb{R}$	k-th component of $\mathbf{x}_i$

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The general scope of this thesis mainly refers to the application of existing kernel-based methods and their subsequent adaptation to learning problems with low number of data points and high dimensional number of measured variables. Kernel learning methods and regularization techniques cover a wide range of advanced and powerful mathematical techniques within a sound statistical framework. These methodologies attempt to view the learning problem using general building blocks whose flexibility allows the construction of non-linear models for classification, regression, feature extraction, variable selection and clustering among others. Our main interest lies in developing efficient, yet simple, algorithms to deal with few number of data points, large number of measured variables, partially labeled data points and large-scale clustering. Such type of data sets arise very frequently from biological systems and high-throughput technologies.

Active research on regularized kernel methods during the last decade has brought together concepts from statistics and optimization theory, into a large variety of successful applications in areas such as medicine, biology, economics and earth sciences. In particular, the joint work of molecular biology, engineering and computer science communities has given place to a new emerging field: *bioinformatics*. The term bioinformatics refers to the development of algorithms, computational and statistical techniques as

well as mathematical theory to solve formal and practical problems inspired from the management and analysis of biological data. In particular, the development of microarray technologies has urged more and more the demand for efficient algorithms and robust data analysis. Microarray chips are capable of measuring the expression levels of thousands of genes simultaneously. Parallel measurements of these gene expression levels result in data vectors containing thousands of values, referred as expression patterns. Likewise, mass spectrometry technologies generate large amounts of data measuring the activity and/or expression of thousands of proteins, on a given set of biological samples. Moreover, mass spectral imaging (MSI) adds yet another dimension to the data structure by preserving the spatial information and therefore introducing a valuable source of prior information that should not be disregarded but rather incorporated into the learning problem. There are, nevertheless, several common problems to all these high-throughput technologies and they relate to: large number of variables, low number of examples, low signal-to-noise ratio, irrelevant variables and the presence of missing values and outliers. For this purpose, methods and algorithms capable of handling high-dimensional data vectors and capable of working under a minimal set of assumptions are required. Through this work, we attempt to propose alternatives to tackle the mentioned issues.

To a large extent, this thesis is about Support Vector Machine (SVM) type classifiers [111] and, more generally Regularized Least Squares Models, which include a larger family of techniques. A particular class of kernel machines is the least squares support vector machines (LS-SVM) [100, 101], which simplifies the classical SVM formulation. Besides, LS-SVMs cover a wide range of learning algorithms that can be formulated within a primal-dual framework. The development of algorithms and mathematical models that can learn from data involves, however, careful parameter tuning and complexity control of the statistical models. Throughout this work, model selection will play a central role in the construction of reliable and reproducible algorithms. We focus primarily on the use of regularized linear models and kernel based techniques and orient our approach from the algorithm's design point of view, before moving gradually to customized learning tasks involving either microarray or mass spectrometry data.

Consequently, the general objective in this work concerns the application of existing kernel-based methods to particular problems in the areas of microarray and (imaging) mass spectrometry data analysis. First, we consider the problem of selecting relevant genes for prediction in microarray data analysis, widely acknowledged as the variable selection problem. Secondly, we deal with the incorporation of prior knowledge from MSI experiments and semi-supervised learning for partially labeled data. Finally, we address the question of clustering similar genes through non-linear clustering algorithms for large scale microarrays. Overall, we touch on the topics of clustering, prediction and classification models, variable selection and incorporation of prior knowledge.

1.2 Challenges

The current challenges and problems are described as follows:

- **Microarray data:** A common characteristic in “omics" data is the high dimensionality and relatively low number of data points. Microarray and mass spectrometry technologies are designed to understand how the whole set of genes and proteins of a particular biological sample is functioning. Indeed, on a single microarray experiment, thousands of spots, each related to a single gene, are used to simultaneously detect gene expression levels while varying several experimental conditions [16]. Parallel measurements of these expression levels result in data vectors that contain thousands of values. These data vectors are called expression patterns. Traditional data analysis techniques cannot directly deal with high dimensional data vectors and a pre-processing step is usually needed. Therefore, the new direction is put into methods and algorithms capable of handling high dimensional data vectors and work under a minimal set of assumptions.

- **Mass spectrometry data:** Due to modifications and regulation of biologically active molecules, microarrays do not capture all relevant phenomena in the cell at the molecular level. Hence, the study of the *proteome* (analogous term to *genome* for the global set of proteins) allows to obtain additional information about the molecular biology of tissues and samples. A proteomics study typically tries to capture a snapshot of the current protein content in the cell. In this setting, selective separation

techniques such as liquid chromatography, gel electrophoresis and mass spectrometry represent the principal techniques behind high-throughput proteomics. In particular, mass spectrometry (MS) has become the central driving force to quantify the presence of a large subset of proteins in a given sample. Within the MS category, mass spectral imaging additionally provides extra information about the spatial distribution of biomolecules in the organic tissue. Quantitatively, these technologies deliver mass spectra containing thousands of discrete peak intensities, each associated with a mass-to-charge (m/z) value, which in turn is associated with a (unknown) protein or a fragment thereof. The final data output, similarly to microarray data, consists of huge data vectors, where each component quantifies the presence in amount of an unspecified protein (or fragment) in the sample at hand.

- **Variable selection:** As mentioned earlier, the high dimensionality and low number of data points in typical microarray or mass spectrometry experiments, let alone the likely presence of irrelevant measurements, demand efficient mechanisms to select informative genes or mass-to-charge regions. Such problem can be seen as finding the needle(s) in a huge haystack. Variable selection techniques aim at finding a subset of variables that results in more accurate and compact set of predictors. The goal is then to filter out those inputs that are irrelevant to the specific model. However, there is no single definition for *the best* subset and different algorithms will produce different subsets. In practice, one needs to define (i) how to search in the space of all possible variable subsets and (ii) how to assess the prediction performance of the learning machine. We shall consider the LS-SVM learning algorithm as a black box to score subsets of variables according to their predictive power.

- **Model selection:** In practice, the chosen statistical model might be too complex for the application at hand and thus over fitting phenomena can easily occur. For this reason, model and parameter selection are key issues in the design of learning models. In this context, regularization techniques and disciplined parameter selection is of utmost importance. Resampling techniques such as leave-one-out (LOO) and cross-validation are standard approaches to select parameters and assess the model's complexity. We emphasize in the use of fast and efficient resampling techniques during the model building phase.

- **Lack of labeled data:** The labor of manually labeling data is not a trivial task. Besides, not only is manual labeling time consuming and expensive but it might also be highly sensitive to human error. The label completion task consists then in using effectively a small amount of *a priori* labeled data points, for instance tissue regions on a biological sample, while providing a way to propagate or spread these labels over the unmarked regions. This task is usually accomplished through semi-supervised methodologies, in which the labeled data points constitute side information and impose a prior structure for the remaining unlabeled data points. The final labeling should be consistent with respect to the initial imposed restrictions.
- **Incorporating prior information:** In real-life applications, additional side information is typically available together with the data. This prior information might come from expert knowledge, inherent data structure, or even redundancy. Therefore, it becomes important to have a flexible methodology as to incorporate this extra source of information in a systematic manner. For instance, the spatial distribution present in MSI experiments suggests another information component and should not be disregarded.
- **Large-scale problems:** During the last 10 years, the amount of data available being produced in areas such as bioinformatics, text mining, computer vision, information retrieval, process industry has been increasing at exponential rate. Large-scale problems become a challenge for any type of machine learning technique. In order to extract useful information from this tsunami of data, approximation techniques, reduced set methods, subsampling schemes and sparse models are required to deal with the large-scale data while using standard hardware.

1.3 Goals of this thesis

The objectives of this thesis are summarized as follows:

1. Extend the core LS-SVM methodologies to cope with variable selection in high dimensional settings. Not only should the algorithms be efficient, but they should also include mechanisms for model selection. A full methodology integrating variable selection, parameter selection within a single efficient framework represents a major milestone through this work.

2. Adapt and exploit the structure of the LS-SVM classifiers as to efficiently update the model parameters, such that these algorithms can be employed with microarray data sets.

3. Incorporate additional side information from mass spectrometry and mass spectral imaging mass spectrometry. Sources of information refer to the high redundancy at the m/z level and the spatial location within the biological sample. Mass spectrometry data reveals a highly sparse structure and consequently large portions of the m/z domain do not provide any contribution. We aim at building sparse models that can disregard the contribution of these m/z values.

4. Use the partially labeled regions in MSI experiments in conjunction with semi-supervised models as to complete the unlabeled tissue regions and differentiate among them based on their m/z signature. By considering the spatial relationships and proximity of spectra within the tissue, we attempt to model the spatial distribution of the spectra.

5. Develop kernel algorithms for clustering capable of handling large-scale microarray data sets. The clustering algorithm should incorporate mechanisms to select a reduced number of genes and infer the cluster membership of the left-out genes. Additionally, a hierarchical structure will help elucidate subtle clusters in this large scale setting, that are otherwise obfuscated by the presence of large prominent clusters.

1.4 Chapter by Chapter Overview

This thesis is organized in six main chapters. Figure 1.1 visually illustrates the structure of this work. The content of the chapters is organized as follows:

- Chapter 2 provides general introduction to regularized methods for learning as well as kernel-based methods. We present applications to microarray and mass spectrometry data sets using the LS-SVM method and highlight its benefits. Of particular interest is the application of Transcription Factor (TF) prediction which requires training of multiple LS-SVM in order to infer the missing information of this ill-posed task. Effective model selection and algorithms' efficiency favor LS-SVMs.
- Chapter 3 presents LS-SVM formulations for the problem of variable selection. To overcome the computational burden of the variable selection problem, we develop algorithms using LS-SVM classifiers with linear kernels. The model parameters are efficiently updated at every stage by using low rank matrix updates. The proposed approach is applied to several benchmark data sets as well as two microarray data sets, comparing favorably to existing methodologies.
- Chapter 4 extends the work presented in Chapter 3 to more general kernels. Starting from component-wise (additive) models, we construct non-linear algorithms for variable selection. In the case of component-wise polynomial kernels, we demonstrate that a finite-dimension feature map yields low rank matrices which, as in the case of the linear kernels, can be efficiently updated.
- Chapter 5 describes a learning approach for Mass Spectral Imaging (MSI) data. Taking into account the special structure of MSI data, we devise sparse predictive models that simultaneously perform variable selection in the spirit of ℓ_1 penalized formulations. We explore the use of a smooth quadratic penalty to model the natural ordering of the physical variables of MSI data, that is, the ordered sequence of m/z values. To overcome the lack of labeled data, we model the spatial proximity among spectra by means of a connectivity graph over the set of predicted labels.
- Chapter 6 covers the use of spectral clustering and out-of-sample extension as first proposed in [9]. This approach offers the possibility to train and validate the clustering model within the standard learning

setting of supervised models [8, 9]. We make use of this algorithm to cluster genes in groups based on their expression values. Firstly, the clustering model is built upon a selected subset of informative genes and, in a second stage the model is used to infer cluster memberships for the remaining genes. Informative genes are selected via entropy maximization, related to the underlying density distribution of the data set. This subsampling scheme greatly reduces the computational burden for large number of genes while the out-of-sample extension provides a clustering model for new data points. In addition, we provide a hierarchical implementation that allows to refine the clustering partitions at different levels of resolution.

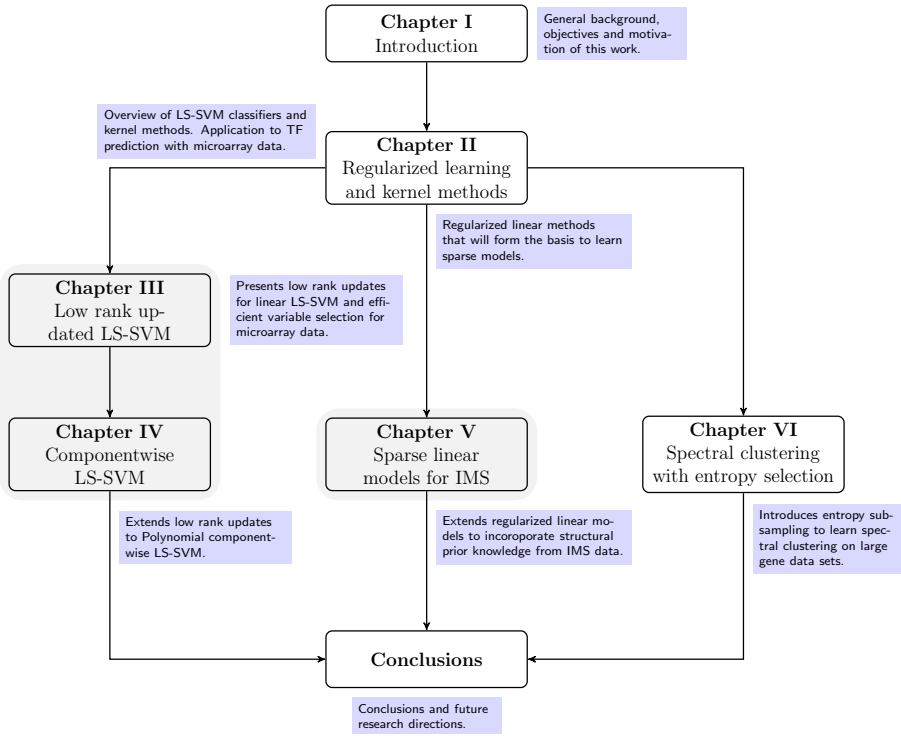


Figure 1.1: Structure of the thesis. Arrows suggest the reading order of the chapters. Chapters III, IV and V constitute the main contributions of this thesis.

1.5 Contributions of this thesis

The main contributions of this thesis are summarized as follows:

- **Efficient variable selection**. The use of LS-SVM classifiers as learning algorithm, is motivated on the basis of its special structure. The leave-one-out (LOO) estimator, directly obtained as a by-product of the basic training algorithm [19, 20, 109], is taken as the basis of the variable selection criterion. Moreover, considerable savings in terms of computational complexity can be achieved with a particular case of low rank modifications on the LS-SVM solution. Simplifications that eventually allow the practical implementation of forward and backward search strategies even for a relatively high number of variables [74, 75].
  - F. Ojeda, J.A.K. Suykens, and B. De Moor. Variable selection by rank-one updates for least squares support vector machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'07)*, pages 2283–2288, Orlando, U.S.A., 2007.
  - F. Ojeda, J.A.K. Suykens, and B. De Moor. Low rank updated LS-SVM classifiers for fast variable selection. *Neural Networks*, 21(2-3):437–449, 2008.
- **Non-linear variable selection and component-wise models**. Using the framework of Low-ranked updated LS-SVM, we present an extension to more general type of kernels. We start from additive/componentwise kernels that in the case of polynomial kernels yield low rank matrices. Based on this observation, we develop algorithms for sequential variable selection that can be efficiently trained and updated [72].
  - F. Ojeda, T. Falck, B. De Moor, and J.A.K Suykens. Polynomial componentwise LS-SVM: fast variable selection using low rank updates. In *International Joint Conference on Neural Networks (IJCNN 2010)*, pages 3291–3297, July 2010.
- **Sparse linear models for MSI**. We address issues regarding inherent ordering of the model variables and the spatial relationships of the data points. In a first step, we start from regularized models that impose sparsity on the solution of coefficients. In the problem of interest, variables possess a natural ordering due to their physical meaning.

Therefore, we enforce that the estimated coefficients of nearby variables should smoothly vary in terms of m/z.

- **Incorporation of prior knowledge**. We develop semi-supervised models that use the labeled portions of the tissue coming from MSI experiments to help predict the anatomical labels of the unlabeled regions of the tissue. The medical objective of such models would be, for instance, to provide the pathologist with insight in interpreting molecular tissue content of areas that do not lend themselves for straightforward human classification.

- **Partially labeled data and semi-supervised learning**. Our approach encodes the spatial proximity among spectra by means of a graph and hence can be seen as a semi-supervised method. The resulting proposed model is shown to be equivalent to a *LASSO* formulation. Each component in our optimization problem clearly embodies the structural information of MSI data, whereas regularization parameters trade off the complexity of the model in terms of sparsity, smoothness and unlabeled data points [73].

  – F. Ojeda, M. Signoretto, R. Van de Plas, E. Waelkens, B. De Moor, and J.A.K Suykens. Semi-supervised learning of sparse linear models in mass spectral imaging. In T. Dijkstra, E. Tsivtsivadze, E. Marchiori, and T. Heskes, editors, *Pattern Recognition in Bioinformatics*, volume 6282 of *Lecture Notes in Computer Science*, pages 325–334. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-16001-1_28.

- **Spectral clustering and entropy based selection**. We consider the problem of subset selection in spectral clustering with application to large set of genes coming from microarray experiments. Starting from a small subset of genes actively selected via entropy maximization, the spectral clustering model is first trained and then validated. Once the final cluster model is provided, the remaining data points are assigned using the out-of-sample extension method. This subsampling scheme greatly reduces the computational burden in large-scale gene expression data while the out-of-sample extension provides a clustering model for new data points. Moreover, in order to detect subtle clusters in large scale data sets and therefore avoid forming oversimplified clusters, we adapt

our methodology to work in a hierarchical fashion with clusters being refined at each level [70, 71, 83].

- N. Pochet, F. Ojeda, F. De Smet, T. De Bie, J.A.K Suykens, and B. De Moor. *Kernel clustering for knowledge discovery in clinical microarray data analysis*, chapter III, pages 64–92. Kernel Methods in Bioengineering, Signal and Image Processing. Idea Group Inc., Hershey, Pennsylvania, 2007.
- F. Ojeda, C. Alzate, J.A.K. Suykens, and B. De Moor. A large scale spectral clustering with out-of-sample extension: application to gene expression data. 15th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB) and 6th European Conference on Computational Biology (ECCB), July 2007. Outstanding poster award.
- F. Ojeda, C. Alzate, B. De Moor, J.A.K Suykens. Entropy based selection for spectral clustering. Internal Report 11-83, ESAT-SISTA K.U. Leuven, Leuven, Belgium, 2011.

Chapter 2

Regularized learning and kernel methods

This chapter reviews the main concepts in regularized learning methods and kernel-based learning for classification problems. Starting from the general framework of regularized least squares, we move into concepts about kernel functions and how they allow for a more flexible data representation in high-dimensional spaces. Particularly, the support vector machine (SVM) method for classification is discussed and compared to the LS-SVM (Least-squares support vector machine) formulation. Throughout most of this thesis, the primal-dual formulations of LS-SVM methods play a central role in the construction of predictive models due to their simplicity. To illustrate the flexibility and modeling capabilities of (LS)-SVM, we consider two applications in bioinformatics namely, classification of mass spectrometry from the Clinical Proteomics Program Databank, and transcription factor prediction using microarray and Chip-CHip data sources.

2.1 Ridge regression and the LASSO

In standard prediction problems, one is presented with a set of training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where the input $\mathbf{x}_i \in \mathbb{R}^d$, is described by d variables, and the corresponding output $y_i \in \mathbb{R}$. Usually, one seeks to construct a prediction rule from the training data, so that given a new input $\mathbf{x}_i$, one can estimate an output y_i for it. The goal consists then in estimating a model of the form

$$y_i = \mathbf{w}^\top \mathbf{x}_i + \varepsilon_i \ , \qquad \varepsilon_i \sim \mathcal{N}(0, \hat{\sigma}^2) \ . \tag{2.1}$$

Equivalently, stacking all the data points into a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ one may write

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\varepsilon}. \tag{2.2}$$

The vectors $\mathbf{w} \in \mathbb{R}^d$, $\mathbf{y} \in \mathbb{R}^n$, contain the model weights (to be estimated) and output values respectively. In ridge regression, the weight vector $\mathbf{w} = [w_1, w_2, \ldots, w_j, \ldots, w_d]^\top$, is estimated by minimizing the training error and controlling the norm $\|\mathbf{w}\|_2^2$, that is how large the coefficients may grow below a given upper bound $\eta > 0$. The objective function is formulated as:

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \left(y_i - \mathbf{w}^\top \mathbf{x}_i \right)^2 \tag{2.3}$$

$$\text{s.t.} \sum_{j=1}^{d} w_j^2 \leq \eta \ .$$

Or alternatively, in matrix notation

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

$$\text{s.t. } \mathbf{w}^\top \mathbf{w} \leq \eta \ ,$$

which is equivalent to the optimization problem

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \mathbf{w}^\top \mathbf{w} \ , \tag{2.4}$$

where $\lambda > 0$. This objective function is minimized for

$$\mathbf{w} = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_d \right)^{-1} \mathbf{X}^\top \mathbf{y} \ . \tag{2.5}$$

The first term in (2.4) measures how well the prediction function fits the training data. The second term trades off between the loss on the training set and the complexity of the prediction function. Hence λ is called the regularization parameter. For $\lambda = 0$, the ordinary least squares (OLS) solution is recovered in which the w_j's are left unconstrained. In that case, the weights may take very large values and hence are susceptible to very high variance. Solution (2.5) is indexed by the tuning parameter $\lambda > 0$. Inclusion of λ makes the problem non-singular even if $\mathbf{X}^\top\mathbf{X}$ is not invertible, which is in fact the original motivation for ridge regression with respect to OLS. For each λ, there exists a solution controlling the size of the coefficients and amount of regularization. Ridge regression includes thus all predictors but with smaller coefficients compared to ordinary least squares (OLS).

In contrast to ridge regression, the use of a ℓ_1 penalty does reduce coefficients to zero, thus producing models that are sparse. This approach yields least absolute shrinkage and selection operator (LASSO) [104]. Similarly to regularized least squares, the LASSO algorithm minimizes the sum of squared errors but instead it imposes a fixed bound η on the sum of the absolute values of the regression coefficients.

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \tag{2.6}$$

$$\text{s.t. } \sum_{j=1}^{d} |w_j| \leq \eta \ .$$

where $\eta > 0$. By varying η, LASSO regression gives solutions ranging from sparse estimates of the least squares estimates to the OLS solution ($\eta \to \infty$). Alternatively, one may also write the LASSO problem as in the form of a penalized residual sum of squares:

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \sum_{j=1}^{d} |w_j| \ . \tag{2.7}$$

There exists a $\lambda > 0$ such that solutions to problems (2.6) and (2.7) become equivalent. For all the algorithms so far exposed, one needs a disciplined way of selecting λ, that is, one requires to *tune* or find a suitable value for λ. Although there exist many heuristics to select a suitable value for λ, the standard practice is to use cross-validation. Contrary to ridge regression, there is no closed form

solution to solve (2.7). Nevertheless, a wide variety of techniques ranging from sequential quadratic programming [53], coordinate descent methods [35], and homotopy methods [77]. An interesting approach to solve the LASSO problem is the LARS (Least Angle Regression) algorithm [28]. LARS involves solving a sequence of least-squares problems on an increasingly large subspace, defined by the active set of variables. We explore the use of this algorithm for Imaging Mass Spectrometry in Chapter 5.

2.2 Kernel based learning

2.2.1 Mercer kernels

A positive definite kernel (also called Mercer kernel) on a space $\mathcal{X}$ is defined as a symmetric continuous function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that for any set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathcal{X}$, and any value n, the $n \times n$ kernel matrix $\mathbf{\Omega}$ defined by $\mathbf{\Omega}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j), i, j = 1, \ldots, n$ is positive definite. A Mercer kernel induces a high dimensional feature space $\mathcal{F}$ and a mapping $\varphi(\cdot)$ from $\mathcal{X}$ to $\mathcal{F}$ such that $K(\mathbf{x}, \mathbf{z})$ equals the dot product between $\varphi(\mathbf{x})$ and $\varphi(\mathbf{z})$. The latter property was introduced in 1909 in Mercer's theorem:

Theorem 2.1. ***Mercer [13]*** *Suppose $\mathcal{X} \subseteq \mathbb{R}^d$ and K is a symmetric continuous function such that the integral operator*

$$T_K : L_2(\mathcal{X}) \to L_2(\mathcal{X})$$

$$(T_K f)(\cdot) = \int_{\mathcal{X}} K(\cdot, \mathbf{x}) f(\mathbf{x}) d\mathbf{x} \ ,$$

is positive, which means that for all $f \in L_2(\mathcal{X})$

$$\int_{\mathcal{X} \times \mathcal{X}} K(\mathbf{x}, \mathbf{z}) f(\mathbf{x}) f(\mathbf{z}) d\mathbf{x} d\mathbf{z} \geq 0 \ .$$

Let $\phi_j \in L_2(\mathcal{X})$ be normalized orthogonal eigenfunctions of T_K with corresponding eigenvalues $\lambda_j > 0$. Then $K(\mathbf{x}, \mathbf{z})$ can be expanded as

$$K(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{\infty} \lambda_j \phi_j(\mathbf{x}) \phi_j(\mathbf{z}) \ . \tag{2.8}$$

For a number of finite terms, d_φ, (2.8) can be written as

$$K(\mathbf{x},\mathbf{z}) = \sum_{j=1}^{d_\varphi} \sqrt{\lambda_j}\phi_j(\mathbf{x})\sqrt{\lambda_j}\phi_j(\mathbf{z}) = \varphi(\mathbf{x})^\top \varphi(\mathbf{z}) \ , \tag{2.9}$$

where $\varphi(\mathbf{x}) = \left[\sqrt{\lambda_1}\phi_1(\mathbf{x}), \sqrt{\lambda_2}\phi_2(\mathbf{x}), \ldots, \sqrt{\lambda_{d_\varphi}}\phi_{d_\varphi}(\mathbf{x})\right]^\top$ represents the feature vector. Thus, a positive definite kernel induces a nonlinear mapping $\varphi(\cdot)$ to a high dimensional feature space $\mathcal{F} \in \mathbb{R}^{d_\varphi}$ (which can be infinite dimensional) without explicitly computing this space. The kernel evaluation provides the dot product of the mapped points in $\mathcal{F}$.

Consider as a simple example, computing the Euclidean distance of the points $\mathbf{x}$ and $\mathbf{z}$ in a kernel-induced feature space: $||\varphi(\mathbf{x}) - \varphi(\mathbf{z})||_2$. Expanding the norm, one can write $\left(\varphi(\mathbf{x})^\top\varphi(\mathbf{x}) - 2\varphi(\mathbf{x})^\top\varphi(\mathbf{z}) + \varphi(\mathbf{z})^\top\varphi(\mathbf{z})\right)^{1/2}$ and applying (2.9) leads to $\left(K(\mathbf{x},\mathbf{x}) - 2K(\mathbf{x},\mathbf{z}) + K(\mathbf{z},\mathbf{z})\right)^{1/2}$. The use of (2.9) for interpreting kernels as dot products in a feature space was coined as *the kernel trick* and introduced into pattern recognition in 1964 [2]. After the SVM, other classical linear algorithms such as PCA, Fisher discriminant analysis (FDA), k-means, and ridge regression have been also *kernelized*.

2.2.2 Kernel functions

Typical choices of positive definite kernel functions when $\mathcal{X} \subseteq \mathbb{R}^d$ are:

- Linear kernel:

$$K(\mathbf{x},\mathbf{z}) = \mathbf{x}^\top\mathbf{z} \ . \tag{2.10}$$

- Polynomial kernel:

$$K(\mathbf{x},\mathbf{z}) = (\mathbf{x}^\top\mathbf{z} + \tau)^p \ , \tag{2.11}$$

  where $p \in \mathbb{N}$ is the degree and $\tau \geq 0$. The feature space induced by the polynomial kernel has dimensionality $\binom{d+p}{p}$ [111].

- Radial Basis Function (RBF) kernel:

$$K(\mathbf{x},\mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|_2^2/\sigma^2) \ , \tag{2.12}$$

  where $\sigma \in \mathbb{R}$ is the kernel width. The feature space induced by the RBF kernel is infinite-dimensional for every value of σ [111].

Remark 2.1 (Parameter tuning)**.** *The kernel parameters (e.g. τ, p, σ) should be chosen carefully in order to ensure good generalization. This is particularly important for very flexible kernels such as the RBF kernel where too small values of σ will result in ill-conditioning and over fitting. On the other hand, very large values of σ will make the RBF kernel function behave like a constant function and hence providing no information.*

Remark 2.2 (Normalized kernels)**.** *Consider the cosine of the angle of the points $\mathbf{x}$ and $\mathbf{z}$ in the input space: $\cos(\theta) = \mathbf{x}^\top \mathbf{z}/(||\mathbf{x}||_2||\mathbf{z}||_2)$. Performing this operation in the feature space leads to:*

$$\cos(\theta_\varphi) = \frac{\varphi(\mathbf{x})^\top \varphi(\mathbf{z})}{||\varphi(\mathbf{x})||_2||\varphi(\mathbf{z})||_2} = \frac{K(\mathbf{x},\mathbf{z})}{\sqrt{K(\mathbf{x},\mathbf{x})K(\mathbf{z},\mathbf{z})}} = K_n(\mathbf{x},\mathbf{z}) \ ,$$

where $K_n(\mathbf{x},\mathbf{z})$ denotes a normalized kernel. Normalizing the kernel matrix can be achieved via

$$\boldsymbol{\Omega}_n = \mathbf{M}_n \boldsymbol{\Omega} \mathbf{M}_n \ ,$$

where $\mathbf{M}_n = diag([1/\sqrt{\boldsymbol{\Omega}_{11}}, 1/\sqrt{\boldsymbol{\Omega}_{22}}, \ldots, 1/\sqrt{\boldsymbol{\Omega}_{nn}}])$. Normalized kernels play a key role for robustness in the context of robust statistics.

2.2.3 Support Vector Machines for Classification

Support vector machines [111], provide a solution to the binary classification problem from the statistical learning theory viewpoint. The algorithm separates the two classes by a hyperplane that maximizes the distance between the hyperplane and the nearest data point of each class. Finding such a hyperplane involves the solution of a convex quadratic programming (QP) problem [15]. In the SVM algorithm, the inputs $\mathbf{x}_i$, are first mapped to a high (and possibly) infinite dimensional feature space through the $\varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d_\varphi}$, and the separating hyperplane is constructed in this new feature space (see Figure 2.1).

The convex primal problem is formulated as [111]:

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_{i=1}^{n} \xi_i \\ &\text{s.t. } y_i(\mathbf{w}^\top \varphi(\mathbf{x}_i) + b) \geq 1 - \xi_i \ , \\ &\quad \xi_i \geq 0 \ , \ i = 1, \ldots, n \ , \end{aligned} \tag{2.13}$$

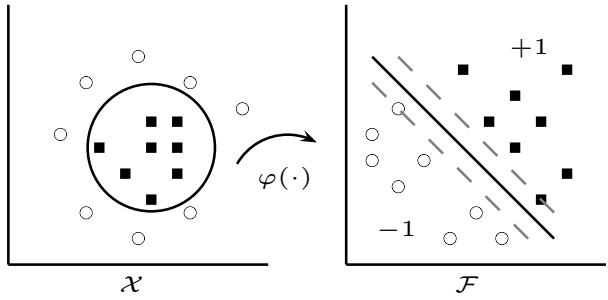


Figure 2.1: *Left*. Kernel trick allows SVMs to map the data from the input space $\mathcal{X}$ to a high-dimensional feature space $\mathcal{F}$ where a linear separation is obtained. *Right*. SVMs then construct a hyperplane that maximizes the margin.

where $C \in \mathbb{R}^+$ is a penalty (cost) parameter for the training error. The slack variables $\xi_i \geq 0$, represent penalization for the misclassified data points. The dual formulation is summarized by the following lemma.

Lemma 2.1. ***[112]*** *Given a positive definite kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with $K(\mathbf{x}, \mathbf{z}) = \varphi(\mathbf{x})^\top \varphi(\mathbf{z})$ and regularization constant $C \in \mathbb{R}^+$, the dual problem to (2.13) is given by:*

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.14}$$

$$\text{s.t.} \sum_{i=1}^{n} \alpha_i y_i = 0 \ , \ 0 \leq \alpha_i \leq C \ , \ i = 1, \ldots, n \ .$$

Proof. The Lagrangian of problem (2.13) is

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \xi_i; \alpha_i, \nu_i) = & \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^{n} \xi_i + \ldots \\ & - \sum_{i=1}^{n} \alpha_i (y_i (\mathbf{w}^\top \varphi(\mathbf{x}_i) + b) - 1 + \xi_i) - \sum_{i=1}^{n} \nu_i \xi_i \ , \end{aligned} \tag{2.15}$$

where $\alpha_i \geq 0$ and $\nu_i \geq 0$ are the Lagrange multipliers. The solution is given by the saddle point of the Lagrangian:

$$\max_{\alpha_i,\nu_i} \min_{\mathbf{w},b,\xi_i} \mathcal{L}(\mathbf{w}, b, \xi_i; \alpha_i, \nu_i) \ . \tag{2.16}$$

One obtains

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \displaystyle\sum_{i=1}^{n} \alpha_i y_i \varphi(\mathbf{x}_i) \\ \dfrac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \displaystyle\sum_{i=1}^{n} \alpha_i y_i = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \xi_i} = 0 \rightarrow 0 \leq \alpha_i \leq C, i = 1, \ldots, n \end{cases} \tag{2.17}$$

Expressing the problem in terms of α_i and applying the kernel trick $\varphi(\mathbf{x}_i)^\top \varphi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$, leads to the convex QP problem in (2.14). □

The classifier for a new input $\mathbf{x}$, takes now the form

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n_{\text{SV}}} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \right) \ , \tag{2.18}$$

where $n_{\text{SV}} \leq n$ is the number of *support vectors* which are training points with corresponding nonzero α_i. The sparseness property follows from solving the QP problem in (2.14). A number of interesting properties of the SVM algorithm are summarized as follows:

- ⋆ *Global solution*. The solution to the convex QP problem in (2.14) is global and unique when a positive definite kernel is used. For a positive semi-definite kernel the solution is global but not necessarily unique.
- ⋆ *Sparsity*. A number of the resulting Lagrange multipliers, α_i, equals zero. As such, the sum in the in (2.18) should only be taken over all nonzero α_i values, i.e. support values, instead of all data points.
- ⋆ *Geometry*. The corresponding vectors $\mathbf{x}_i$ with nonzero $\alpha_i > 0$, are referred to as the *support vectors*. These data points are located close to the decision boundary and contribute to the construction of the separating hyperplane (see Figure 2.2).

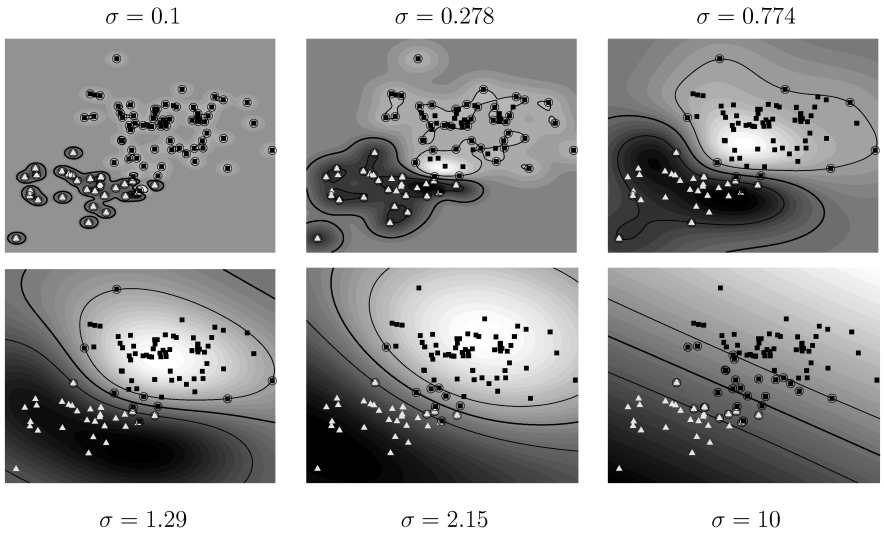


Figure 2.2: The influence of varying the RBF kernel width parameter σ for a fixed value of $C = 10$. From left to right and top to bottom, the parameter σ is changed from 0.1 (top left) to 10 (bottom right). Encircled points represent the support vectors. Increasing σ results in smoother decision boundaries, changing from nonlinear to almost linear.

2.2.4 Least-squares support vector machines

Another class of SVM, are that of the Least-Squares Support Vector Machine (LS-SVM) classifiers [100,101]. These are modifications of the standard support vector machine algorithm and are closely related to regularized least squares (RLS) and ridge regression. With respect to the standard SVM method, the use of equality constraints and an explicit squared loss function in the primal objective. As a result, the solution follows directly from solving a set of linear equations, instead of quadratic programming.

The LS-SVM algorithm considers the following constrained optimization problem [101]

$$\min_{\mathbf{w},b,e} \frac{1}{2}\mathbf{w}^\top\mathbf{w} + \gamma\frac{1}{2}\sum_{i=1}^{n} e_i^2 \tag{2.19}$$

$$\text{s.t. } y_i = \mathbf{w}^\top\varphi(\mathbf{x}_i) + b + e_i \ , \ i = 1,\ldots,n \ ,$$

where $\gamma \in \mathbb{R}^+$ is a regularization parameter controlling the bias-variance trade-off. The variables $e_i \in \mathbb{R}$, are viewed as normal distributed errors of the outputs y_i. Solution to (2.19) is formalized in the following lemma.

Lemma 2.2. ***[101]*** *Given a positive definite kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with $K(\mathbf{x},\mathbf{z}) = \varphi(\mathbf{x})^\top\varphi(\mathbf{z})$ and regularization constant $\gamma \in \mathbb{R}^+$, the dual problem to (2.19) is given by the following set of linear equations:*

$$\left[\begin{array}{c|c} \boldsymbol{\Omega} + \gamma^{-1}\mathbf{I}_n & \mathbf{1} \\ \hline \mathbf{1}^\top & 0 \end{array}\right]\left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline b \end{array}\right] = \left[\begin{array}{c} \mathbf{y} \\ \hline 0 \end{array}\right] \ . \tag{2.20}$$

where $\mathbf{y} = [y_1;\ldots;y_n]$, $\boldsymbol{\alpha} = [\alpha_1;\ldots;\alpha_n]$ and $\boldsymbol{\Omega}_{ij} = K(\mathbf{x}_i,\mathbf{x}_j)$.

Proof. The Lagrangian of problem (2.19) is

$$\mathcal{L}(\mathbf{w},e_i,b;\alpha_i) = \frac{1}{2}\mathbf{w}^\top\mathbf{w} + \gamma\frac{1}{2}\sum_{i=1}^{n} e_i^2 - \sum_{i=1}^{n}\alpha_i(\mathbf{w}^\top\varphi(\mathbf{x}_i) + b + e_i - y_i), \tag{2.21}$$

where $\alpha_i \in \mathbb{R}$ are the Lagrange multipliers. The conditions for optimality are given by

$$\begin{cases} \dfrac{\partial\mathcal{L}}{\partial\mathbf{w}} = 0 \to \mathbf{w} = \displaystyle\sum_{i=1}^{n}\alpha_i\varphi(\mathbf{x}_i) \\ \dfrac{\partial\mathcal{L}}{\partial e_i} = 0 \to \alpha_i = \gamma e_i, i = 1,\ldots,n \\ \dfrac{\partial\mathcal{L}}{\partial b} = 0 \to \displaystyle\sum_{i=1}^{n}\alpha_i = 0 \\ \dfrac{\partial\mathcal{L}}{\partial\alpha_i} = 0 \to \mathbf{w}^\top\varphi(\mathbf{x}_i) + b + e_i - y_i = 0, i = 1,\ldots,n. \end{cases} \tag{2.22}$$

Eliminating the primal variables, $\mathbf{w}, e_i$, leads to

$$\begin{cases} \dfrac{\alpha_i}{\gamma} + \displaystyle\sum_{j=1}^{n} \alpha_j \varphi(\mathbf{x}_j)^\top \varphi(\mathbf{x}_i) + b = y_i, \ i = 1, \ldots, n \\ \displaystyle\sum_{i=1}^{n} \alpha_i = 0 \ , \end{cases} \tag{2.23}$$

With the application of Mercer's theorem, or the kernel trick $\varphi(\mathbf{x}_i)^\top \varphi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$, it is possible to form the kernel matrix $\mathbf{\Omega}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ and express the above equations in matrix notation as to obtain the linear system in (2.20). □

For a new given input $\mathbf{x}$, the least-squares support vector machine classifier is given by

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b \right) \ . \tag{2.24}$$

The main characteristics of the LS-SVM method are summarized as follows:

- ⋆ *Linear system*. Optimal parameters, $\boldsymbol{\alpha}, b$, can be obtained with a computational complexity of $\mathcal{O}(n^3)$ operations [39].
- ⋆ ℓ_2 *cost function*. While in classical SVM many support values are zero (nonzero values correspond to support vectors) in LS-SVM, the support values are proportional to the errors.
- ⋆ *Ridge regression*. A support vector interpretation of ridge regression was proposed for the function estimation problem considering equality type constraints. [90]

The empirical study presented in [110], shows that LS-SVM classifiers achieve comparable performance to standard SVM. Note that the size of the kernel matrix in both the QP problem (2.14) and the system of linear equations (2.20) grows with the number of training data points n. On the other hand, they are independent of the dimension d of the input space. This property makes the dual formulations more convenient for applications with large number of variables.

Remark 2.3 (Primal and dual representations)**.** *In case the feature map $\varphi(\cdot)$ is finite dimensional and explicitly known one has the choice between solving the primal or the dual problem. However, for the RBF kernel on the other hand one can only solve the dual. Consider, for instance, the case of the linear parametric regression model $y = \mathbf{w}^\top \mathbf{x} + b$ with $\mathbf{w} \in \mathbb{R}^d$. The dual representation of the linear model is $y = \sum_{i=1}^n \alpha_i \mathbf{x}_i^\top \mathbf{x} + b$, with $\boldsymbol{\alpha} \in \mathbb{R}^n$. One distinguishes then between the following cases:*

- *Case d small, n large: solving the primal problem in $\mathbf{w} \in \mathbb{R}^d$ is more convenient.*
- *Case d large, n small: solving the dual problem in $\boldsymbol{\alpha} \in \mathbb{R}^n$ is more convenient.*

Therefore within a setting of primal and dual model representations, the modeling approach can be tailored towards the given data problem. While kernel methods offer flexible representations in the dual, parametric interpretations preserve the global picture in the primal. In Chapter 3, this dual view is further exploited for variable selection with LS-SVM and linear kernels in problems with large number of input variables.

2.3 Variable selection

In general, the major aims of variable selection for classification are finding a subset of input variables that result in more accurate classifiers and therefore more compact models. An amount of variables can be redundant to the classification problem and thus can variable selection filter out those variables irrelevant for the model. Three major types of variable selection methods are distinguished, namely filter, wrapper and embedded methods [3, 54, 103, 115].

- *Filter methods* consider variables independently of the learning algorithm. A simple filtering technique ranks or scores each variable based on some measures like information gain criterion, mutual information, or t-tests. Since the learning algorithm (e.g. the classifier) is not involved, their main advantage is the low computational cost [54]. Ignoring the impact of the learning algorithm, however, constitutes their weakest point.

- *Wrapper methods* [54], take into account different subsets of variables that are tested using the learning algorithm. The performance of each subset is measured using a *scoring function* like n-fold cross-validation or leave-one-out cross-validation. Since the size of the search space scales with the number of variables as 2^d, exhaustive search through the input space unfeasible. Therefore, heuristic search methods such as *backward elimination* or *forward selection* are often used. Backward elimination starts from all variables and consecutively removing the least significant, until all variables in the model satisfy a level of significance. Forward selection includes variables until a level of significance is reached. In general wrapper methods provide higher accuracy, their disadvantage is the high computational cost of the search nevertheless [54].

- *Embedded methods* aim to integrate the variable selection within the learning algorithm. Gradient-based algorithms are typically used in minimizing generalization error bounds. For instance in [114] the radius-margin bound for SVM is optimized with respect to individual variable scaling factors, while in [22] derivations for several SVM generalization bounds in terms of the variable scaling factors are provided. Within this category automatic relevance determination (ARD), aims to identify informative input variables as a natural result of optimizing the model selection criterion [22, 86]. In general, these methods use an elliptical RBF kernel $K(\cdot,\cdot)$ that depends on d tuning parameters $\boldsymbol{\theta} = [\theta_1 ... \theta_d]^\top$. The decision function is thus parametrized by α_i, b and $\boldsymbol{\theta}$ and takes the form

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}_i) + b\right) \quad ,$$

  where the k-th component of $\mathbf{x}$ is denoted by $x^k \in \mathbb{R}$ with

$$K_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = \exp\left(-\sum_{k=1}^{d} \theta_k \left(x^k - z^k\right)^2\right) \quad . \tag{2.25}$$

  Partial derivatives with respect to the kernel parameters provide the necessary information to perform gradient-based optimization. In this setting upper bounds on the LOO error (e.g. radius-margin bound) can be minimized to optimize the kernel parameter assigned to each variable. Large θ_k's indicate relevant variables.

2.4 Model selection and Leave-one-out estimator

The leave-one-out error is referred to as an almost unbiased estimator of the generalization error and it is frequently used to estimate the performance of a learning algorithm [56]. The basic procedure consists in training the learning algorithm from $n-1$ examples, testing the remaining one (leave-out data point) and repeating until all elements have served as test example. The leave-one-out error is defined as

$$\text{error} = \frac{1}{n}\sum_{i=1}^{n} \ell(f^{(-i)}(\mathbf{x}_i), y_i) \ , \tag{2.26}$$

where $f^{(-i)}(\mathbf{x}_i) = \hat{y}_i$ is the prediction made for $\mathbf{x}_i$ by the classifier when trained on all the examples except $\mathbf{x}_i$. The loss function $\ell(\cdot)$ weights the cost (error) when $f(\mathbf{x})$ is predicted instead of y. A typical loss used in classification settings is the so-called *zero-one* function

$$\text{error}_{0/1} = \frac{1}{n}\sum_{i=1}^{n} \Psi\left(y_i - \hat{y}_i^{(-i)}\right) \ , \tag{2.27}$$

where $\Psi(x) = 1, x \geq 0$ is the step function. Alternatively to the step function for the leave-one-out error, continuous functions or upper bounds on the error rate appear to be more tractable for numerical routines. The simplest criterion is the leave-one-out estimate of the sum of squares error, or predicted residual sum of squares (PRESS) statistic, as suggested in [6, 19]

$$\text{error}_{\text{press}} = \frac{1}{n}\sum_{i=1}^{n} \left(y_i - \hat{y}_i^{(-i)}\right)^2 \ . \tag{2.28}$$

Exact computation of (2.26) requires n runs of the learning algorithm. In the case of LS-SVM classifiers, the overall complexity is thus $\mathcal{O}(n^4)$. Such approach becomes quickly prohibitive for a large number of data points. In the context of SVM, particularly, many works have focused in deriving approximations and upper bounds for the leave-one-out error [112]. The main reason is the high computational cost involved when solving n-QP problems.

2.5 Applications

In this section we present applications to microarray and mass-spectrometry data using some of the methods exposed through this chapter. The flexibility of kernel based methods and regularized models is demonstrated through three different tasks. In each of them, model selection plays a crucial role let alone the efficiency of the algorithms for large-scale data sets or high dimensional scenarios.

2.5.1 Classification of mass spectra

Mass spectrometry technologies (SELDI, MALDI) allow to study proteins over a wide range of molecular weights in small biological samples such as serum or tissues. Protein profiles may reflect the pathological changes within an organ of a cancerous subject [81, 82]. Data resulting from MS consist of paired m/z ratio values versus intensities and it is characterized by containing large number of variables and few number of data points (see Figure 2.3). The Clinical Proteomics Program Databank [1] has made publicly available a set of databases for ovarian and prostate cancer. The goal is to build predictive models since the early detection of cancer has the potential to dramatically reduce the mortality associated with this disease.

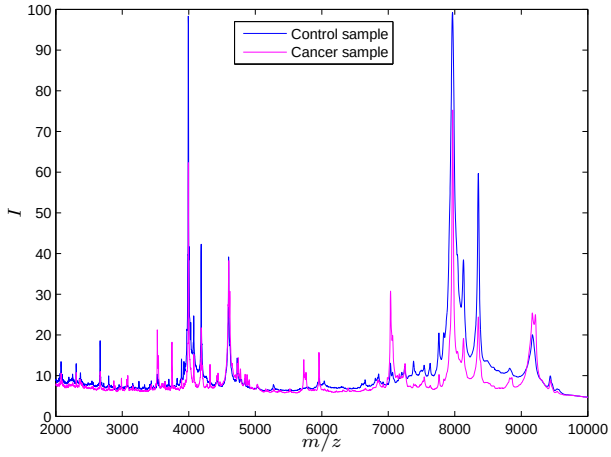


Figure 2.3: Average spectrum of Control and Disease groups [81]

- **Ovarian cancer data set**. This data set consists of serum mass spectra profiles obtained from n_D = 253 subjects, 162 with diagnosed ovarian cancer and 91 non-cancer control subjects [81]. Intensities at 14,554 distinct m/z values were available for analysis. The training subset has been composed by 50 cancer patients and 50 controls, randomly chosen from the complete set.
- **Prostate cancer data set**. This data set corresponds to serum mass spectra profiles from men with a histopathologic diagnosis of prostate cancer (PSA[1] $>$ 4 ng/mL) from those men without prostate cancer (PSA $<$ 1 ng/mL) [82]. A total of n_D = 322 subjects, 69 with histopathologic diagnosis of prostate cancer and 253 control subjects. The training subset is composed by 25 control samples and 31 cancer patients randomly chosen from the initial set.

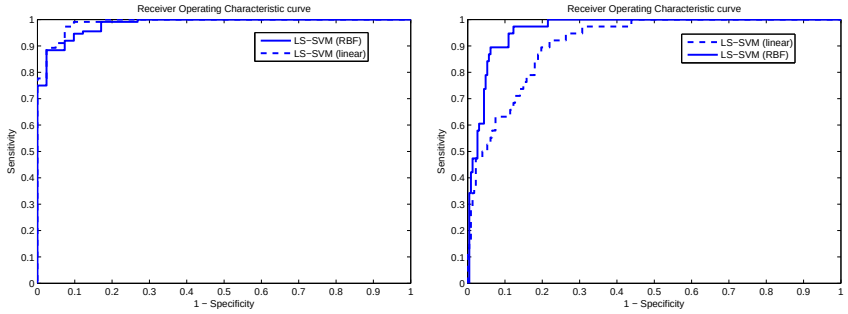
We report classification accuracies for the LS-SVM classifier in Table 2.1. Both linear and RBF kernels are tested with kernel parameters (γ, σ) selected via cross-validation. Results indicate the average over 10 different permutations of the data sets with standard deviation between brackets. Disease (cancer) class is indicated by $y = +1$ whereas $y = -1$ refers to the control class.

Ovarian cancer			Prostate cancer		
LS-SVM (linear)	$\hat{y} = -1$	$\hat{y} = +1$	**LS-SVM (linear)**	$\hat{y} = -1$	$\hat{y} = +1$
$y = -1$	0.995 (0.001)	0.005	$y = -1$	0.799 (0.112)	0.210
$y = +1$	0.006	0.994 (0.002)	$y = +1$	0.100	0.900 (0.037)
LS-SVM (RBF)	$\hat{y} = -1$	$\hat{y} = +1$	**LS-SVM (RBF)**	$\hat{y} = -1$	$\hat{y} = +1$
$y = -1$	0.981 (0.047)	0.019	$y = -1$	0.840 (0.041)	0.160
$y = +1$	0.012	0.988 (0.025)	$y = +1$	0.071	0.930 (0.035)

Table 2.1: Confusion matrix results using 10 different permutations of the test set. Standard deviations are indicated between brackets.

Additionally, the area under receiver operating characteristic curve (AUC) [51] is computed using the LS-SVM predictions on the test data. Figure 2.4(a) compares the AUCs given by LS-SVM classifiers with linear and RBF kernel on the ovarian cancer data. Figure 2.4(b) does it for the prostate cancer data.

[1]PSA: Prostate-specific antigen



(a) (b)

Figure 2.4: Test set ROC for Ovarian and Prostate cancer data sets.

2.5.2 Prediction of transcription factors

Genes working together in certain biological processes are assumed to have similar expression patterns. Therefore, finding similar patterns of genes, having related functions, can deliver evidence to infer the structure of gene networks [36]. Identification of transcription factors (TFs) binding sites in the genome remains incomplete and often difficult solely through chemical techniques. Therefore machine learning approaches constitute a potential alternative with respect to complex experimental methods. Assuming that the full genome of a certain species is given, a particular gene g starts transcription for protein production when a TF t (a protein) chemically binds to it. The machine learning approach to this problem can be summarized as: given a gene g, determine whether TF t binds to it. SVMs can then be used to categorize new genes assuming one provides a set of genes known to be regulated by a certain TF and a set known to be not co-regulated. Figure 2.5 illustrates this concept for a small gene regulatory network.

In this case study microarray data sets, describing the gene expression profiles of yeast under several experimental conditions, are considered [16, 36]. The Spellman data set contains 77 experiments describing the dynamic changes of yeast genes during the cell cycle, while the Gasch data set consists of 177 experiments, examining gene expression behavior during various stress conditions [16, 36]. In order to construct positive and negative data points, a list of known regulation relationships between known TFs and a set of genes is

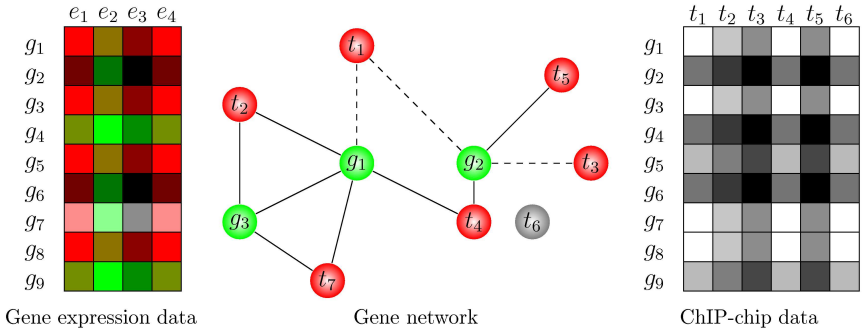


Figure 2.5: Gene regulatory network (GRN) scheme representing known interactions (solid lines) between TF (red circles) and target genes (green circles). Dashed lines represent noninteracting TF-gene pairs. For instance, given the expression levels $[e_1 e_2 \dots e_5]$ of a particular gene g, the learning task aims to determine whether the TF t_6 binds to it. Learning new TF-gene interactions and building gene regulatory networks (center) is based on gene expression data (left) and known TF-gene interactions (right).

required. Such regulatory information is derived from ChIP-chip interactions data collecting binding information of 204 regulators to their respective target genes [43]. This information has previously been used to correlate regulatory programs with regulators and corresponding motifs to a set of co-expressed genes [57]. Values in this regulatory information data set are given in terms of p-values, indicating whether a TF binds to the set of considered genes in yeast. Positive data points are then defined from interactions with a high confidence value, i.e. a p-value ≤ 0.001 [43, 45]. However, the choice of negative data points, i.e. pairs of TF and genes without reported regulatory relationships, is not straightforward since there are few data published establishing that a given TF is found not to regulate a given target gene. Instead of taking a random set of pairs with unknown interaction, negative data points are selected as those genes with a p-value ≥ 0.8 as indicated by the ChIP-chip interaction data. Such genes are less likely to be bound by the TF. After some preprocessing steps (finding common genes and TFs across data sets, removing rows or columns with too many missing values, considering only those TFs with at least 5 known positive interactions), the final expression data consist of a matrix with 5295 genes

$[g_1 g_2 \ldots g_{5295}]$ measured over 250 experimental conditions $[e_1 e_2 \ldots e_{250}]$ and the target matrix with 118 TFs $[t_1 t_2 \ldots t_{118}]$ accounting for about 4000 positive interactions.

Similar to the applications reported in [45,67,84], this case study designs SVM-based classifiers (i.e. LS-SVM with RBF kernel) for each TF. The learning problem is highly unbalanced in all cases with the number of positive data points being outnumbered. Therefore, the following weighted cost function is used for the LS-SVM classifiers

$$\min_{w,e,b} \frac{1}{2}\mathbf{w}^\top \mathbf{w} + \frac{\gamma}{2}\sum_{i=1}^{n} v_i \, e_i^2 \ , \tag{2.29}$$

$$y_i = \mathbf{w}^\top \varphi(\mathbf{x}_i) + b + e_i \ , \tag{2.30}$$

with class specific weights defined as

$$v_i = \begin{cases} \frac{n}{2n_+}, & \text{if } y_i = +1 \ , \\ \frac{n}{2n_-}, & \text{if } y_i = -1 \ , \end{cases} \tag{2.31}$$

and where n_+ and n_-, represent the number of positive and negative data points, respectively. Asymptotically this weighting is equivalent to resampling of the data such that there is an equal number of positive and negative data points.

Training multiple classifiers and tuning their parameters (γ, σ) pose a computational burden, hence, the fast cross-validation implementation for LS-SVM is considered [11, 19]. Figure 2.6 shows the gene regulatory network inferred by the LS-SVM algorithm. Visualization is done using VisAnt software [47]. TF-gene interactions are obtained based on the output of the method by Platt, which is used to assign a probability value based on the LS-SVM output [52]. TFs with at least 5 new interactions with cutoff probability value greater than 0.95 are shown in the regulatory network.

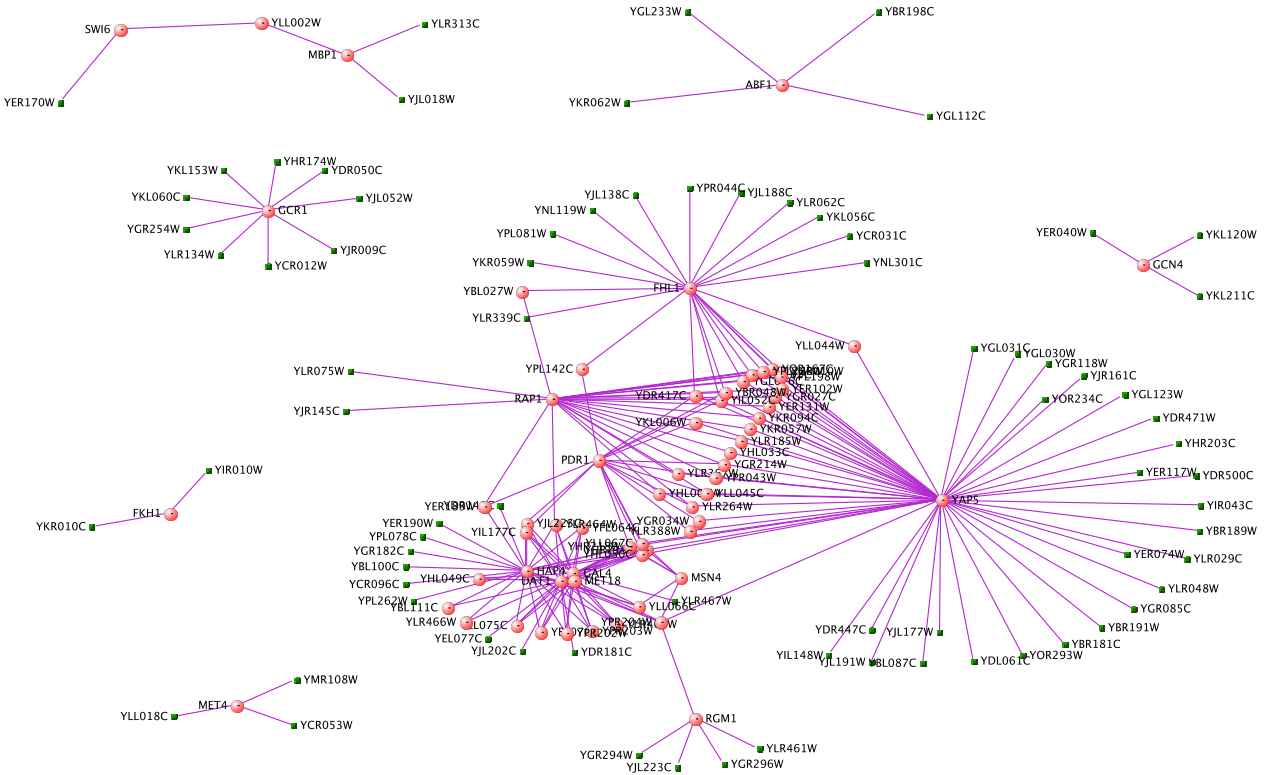


Figure 2.6: Gene network constructed using TFs with at least 5 new predicted interactions by the LS-SVM classifier. The cutoff probability threshold is set to 0.95, such that false positives are less likely to be ranked as new bindings.

Chapter 3

Low rank updated LS-SVM classifiers for variable selection

This chapter presents one of the contributions of this thesis. In order to tackle the computational burden posed by the large number of input variables of the variable selection problem, we develop algorithms using LS-SVM classifiers with linear kernels. The model parameters are efficiently updated at every stage by using low rank matrix updates. In general, the inclusion of a candidate variable into the current model results in a low rank modification to the kernel matrix of the LS-SVM classifiers. In this way, the LS-SVM solution can be updated rather than being recomputed, which improves the efficiency of the overall variable selection process. Relevant variables are selected according to a closed form of the leave-one-out (LOO) error estimator, which is obtained as a by-product of the low rank modifications.

3.1 Variable selection

In the context of classification, variable selection aims at finding a subset of variables that results in more accurate and compact classifiers. Typically, the error of the designed classifier will drop as more variables are added and after reaching some optimal number, it increases again. The goal is then to filter out those inputs that are irrelevant to the specific model. Moreover, the selected variables should not overfit the data. In the ideal case, one would find a unique best subset of variables. However, there is no single definition for *the best* subset and besides different algorithms will return different subsets. In general, variable selection comprises three elements: *(i)* search mechanism, *(ii)* learning algorithm and *(iii)* ranking criterion.

Starting from the set of training data points $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$. Denote by x_i^k the k-th component of $\mathbf{x}_i$. Thus, $\mathbf{x}^k = [x_1^k, x_2^k, \ldots, x_n^k]^\top$ describes the variable k. Thereby, $\mathcal{S} = \left\{\mathbf{x}^1, \ldots, \mathbf{x}^k, \ldots, \mathbf{x}^d\right\}$ is the collection of all d variables. The goal of variable selection is to choose from the full set of all variables $\mathcal{S}$, a reduced subset $\mathcal{S}^* \subset \mathcal{S}, \mathcal{S}^* \in \mathbb{R}^m$, $m < d$, such that an objective function, e.g. cross-validation error, $\mathcal{J}_{\mathcal{S}^*} \leq \mathcal{J}_{\mathcal{S}}$ defined over the set of variables is minimized.

In this chapter we will use the prediction performance of the LS-SVM classifier to assess the relative usefulness of subsets of variables. The learning machine, in fact, is considered as a black box to rank variables according to their predictive power. In practice, one needs to define (i) how to search in the space of all possible variable subsets and (ii) how to assess the prediction performance of the learning machine. We employ therefore both sequential forward selection (SFS) and sequential backward elimination algorithms [54].

3.2 Least squares support vector machine classifiers

From section 2.2.4, we have seen that the dual problem of the LS-SVM method is expressed by the set of $n + 1$ linear equations ([101]):

$$\left[\begin{array}{c|c} \boldsymbol{\Omega} + \gamma^{-1}\mathbf{I}_n & \mathbf{1} \\ \hline \mathbf{1}^\top & 0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline b \end{array}\right] = \left[\begin{array}{c} \mathbf{y} \\ \hline 0 \end{array}\right] \quad , \tag{3.1}$$

where $\mathbf{y} = [y_1; \ldots; y_n]$ and $\boldsymbol{\alpha} = [\alpha_1; \ldots; \alpha_n]$ and $\boldsymbol{\Omega}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. The optimal parameters, $\boldsymbol{\alpha}, b$, can be obtained with a computational complexity of $\mathcal{O}(n^3)$ operations. Alternatively, (3.1) can be factorized into the form of a positive definite system

$$\left[\begin{array}{c|c} \mathbf{H} & 0 \\ \hline 0 & \mathbf{1}^\top \mathbf{H}^{-1} \mathbf{1} \end{array}\right] \left[\begin{array}{c} \boldsymbol{\alpha} + \mathbf{H}^{-1}\mathbf{1}b \\ \hline b \end{array}\right] = \left[\begin{array}{c} \mathbf{y} \\ \hline \mathbf{1}^\top \mathbf{H}^{-1} \mathbf{y} \end{array}\right] \quad , \tag{3.2}$$

Since $\mathbf{1}^\top \mathbf{H}^{-1} \mathbf{1} > 0$, and $\mathbf{H}$ is positive definite, the overall matrix is also positive definite. Model parameters of the LS-SVM system are now given solely in terms of $\mathbf{H}^{-1}$ by

$$b = \mathbf{1}^\top \mathbf{H}^{-1} \mathbf{y} \left(\mathbf{1}^\top \mathbf{H}^{-1} \mathbf{1} \right)^{-1} \tag{3.3}$$

$$\boldsymbol{\alpha} = \mathbf{H}^{-1} \left(\mathbf{y} - b\mathbf{1} \right) \quad . \tag{3.4}$$

This form is very suitable for efficient iterative algorithms involving positive definite matrices [100]. Moreover, this factorization allows us to better describe the efficient LOO estimator in the next section.

3.3 Leave-One-Out estimator for LS-SVM

As described in Section 2.4, the exact computation of the LOO estimator requires n runs of the learning algorithm. Fortunately, for regularized least squares and thus LS-SVM classifiers too, the LOO error can be exactly obtained in closed form with a computational complexity of only $\mathcal{O}(n^3)$ operations. This is equivalent to the time complexity of a single LS-SVM classifier [19, 20, 109]. The systems of linear equations given by (3.1) can be represented in the form of an arbitrary block structured matrix $\mathbf{A} \in \mathbb{R}^{(n+1)\times(n+1)}$ as

$$\left[\begin{array}{c|c} \mathbf{H} & \mathbf{1} \\ \hline \mathbf{1}^\top & 0 \end{array}\right] = \left[\begin{array}{c|c} a_{11} & \mathbf{a}_1^\top \\ \hline \mathbf{a}_1 & \mathbf{A}_1 \end{array}\right] = \mathbf{A} \quad , \tag{3.5}$$

where $\mathbf{a}_1$, is a column vector and a_{11} a scalar. The inverse of a block structured matrix is given by [39]

$$\begin{bmatrix} a_{11} & \mathbf{a}_1^\top \\ \mathbf{a}_1 & \mathbf{A}_1 \end{bmatrix}^{-1} = \begin{bmatrix} \kappa^{-1} & -\kappa^{-1} \mathbf{a}_1^\top \mathbf{A}_1^{-1} \\ -\kappa^{-1} \mathbf{A}_1^{-1} \mathbf{a}_1 & \mathbf{A}_1^{-1} + \kappa^{-1} \mathbf{A}_1^{-1} \mathbf{a}_1 \mathbf{a}_1^\top \mathbf{A}_1^{-1} \end{bmatrix} \quad , \tag{3.6}$$

where $\kappa = a_{11} - \mathbf{a}_1^\top \mathbf{A}_1^{-1}\mathbf{a}_1$. According to [109] and [19], by exploiting the block structure on (3.1), the LOO PRESS error for the i-th training data point can be expressed in a closed form. Given the inverse of $\mathbf{A}$ and the model parameters $\boldsymbol{\alpha}$ solving (3.1) for the full training set are known, the residual of the LOO error for the i-th training data point is [19, 109]:

$$r_i^{(-i)} = y_i - \hat{y}_i^{(-i)} = \frac{\alpha_i}{(\mathbf{A}^{-1})_{ii}} \ . \tag{3.7}$$

Proof. Following [19], let $[\boldsymbol{\alpha}^{(-i)}; b^{(-i)}]$ represent the LS-SVM parameters at the i-th iteration of the LOO procedure. In the first iteration the first training data point is excluded, thus one obtains

$$\begin{bmatrix} \boldsymbol{\alpha}^{(-1)} \\ b^{(-1)} \end{bmatrix} = \mathbf{A}_1^{-1}[y_2, \ldots, y_n, 0]^\top \ .$$

The LOO prediction for the first training data point is then given by

$$\hat{y}_1^{(-1)} = \mathbf{a}_1^\top \begin{bmatrix} \boldsymbol{\alpha}^{(-1)} \\ b^{(-1)} \end{bmatrix} = \mathbf{a}_1^\top \mathbf{A}_1^{-1}[y_2, \ldots, y_n, 0]^\top \ .$$

From the last n equations in (3.1), it is clear that

$$[\mathbf{a}_1 \mathbf{A}_1][\alpha_1, \ldots, \alpha_n, b]^\top = [y_2, \ldots, y_n, 0]^\top \ ,$$

and hence

$$\hat{y}_1^{(-1)} = \mathbf{a}_1^\top \mathbf{A}_1^{-1}[\mathbf{a}_1 \mathbf{A}_1][\boldsymbol{\alpha}^\top, b]^\top = \mathbf{a}_1^\top \mathbf{A}_1^{-1}\mathbf{a}_1\alpha_1 + \mathbf{a}_1^\top[\alpha_2, \ldots, \alpha_n, b]^\top \ .$$

According to the notation in (3.5), the first equation in (3.1) can be stated as $y_1 = a_{11}\alpha_1 + \mathbf{a}_1^\top[\alpha_2, \ldots, \alpha_n, b]^\top$, therefore

$$\hat{y}_1^{(-1)} = y_1 - \alpha_1(a_{11} - \mathbf{a}_1^\top \mathbf{A}_1^{-1}\mathbf{a}_1) \ .$$

From (3.6), $\kappa = a_{11} - \mathbf{a}_1^\top \mathbf{A}_1^{-1}\mathbf{a}_1$, thus

$$y_1 - \hat{y}_1^{(-1)} = \frac{\alpha_1}{(\mathbf{A}_{11})^{-1}} \ .$$

Noting that (3.1) is insensitive to permutations of the ordering of the equations and unknowns, one can write (3.7) $\forall i = 1, \ldots, n$. □

Therefore, the LOO error is now computed using information already available from the training phase of the LS-SVM. This is far more efficient than solving n times the system of linear equations given in (3.1). Given the solution to the basic training algorithm, the complexity of this efficient LOO error computation reduces to $\mathcal{O}(n)$.

Moreover, one can readily notice that only the diagonal elements of $\mathbf{A}^{-1}$ need to be determined. Using (3.6), the inverse of $\mathbf{A}$ can be expressed in terms of $\mathbf{H}^{-1}$

$$\mathbf{A}^{-1} = \left[\begin{array}{c|c} \mathbf{H}^{-1} + \frac{1}{s}\mathbf{H}^{-1}\mathbf{1}\mathbf{1}^\top\mathbf{H}^{-1} & -\frac{1}{s}\mathbf{H}^{-1}\mathbf{1} \\ \hline -\frac{1}{s}\mathbf{1}^\top\mathbf{H}^{-1} & \frac{1}{s} \end{array}\right] \ , \tag{3.8}$$

where the Schur complement $s = -\mathbf{1}^\top\mathbf{H}^{-1}\mathbf{1}$ is a scalar. Since only the first n diagonal elements of $\mathbf{A}^{-1} \in \mathbb{R}^{n+1\times n+1}$ are relevant in computing (3.7) and given that $\mathbf{H}^{-1} \in \mathbb{R}^{n\times n}$ is already known, then using the upper left block in (3.8) one obtains

$$(\mathbf{A}^{-1})_{ii} = (\mathbf{H}^{-1})_{ii} + \frac{\nu_i^2}{s} \ , \ i = 1, \ldots, n \ , \tag{3.9}$$

with $\boldsymbol{\nu} = \mathbf{H}^{-1}\mathbf{1}$, $\boldsymbol{\nu} \in \mathbb{R}^{n\times 1}$. Alternatively, (3.9) can be defined in terms of the Cholesky factorization of $\mathbf{H} = \mathbf{L}\mathbf{L}^\top$ [19]. In summary, the procedure to compute the efficient LOO error is presented in Algorithm (1).

Algorithm 1 Efficient LOO for LS-SVM classifiers. The LOO error is obtained as a by-product after a single run of the LS-SVM classifier.

1: Construct kernel matrix and obtain $\mathbf{H}^{-1}$.
2: Solve for b and $\boldsymbol{\alpha}$ using (3.4).
3: Compute diagonal elements $(\mathbf{A}^{-1})_{ii}$ with (3.9).
4: Compute prediction residuals using (3.7).

To summarize, the LS-SVM classifiers can be efficiently trained by solving just one system of linear equations. Additionally, a simple training of the LS-SVM is sufficient to obtain the LOO estimator with less computational burden than a naive implementation. Such LOO error criterion is used to evaluate whether potential variables must be included or discarded.

Remark 3.1 (Complexity of the naive approach)**.** *Naive computation of the LOO estimator requires n trainings. If the forward selection goes through $\mathcal{O}(d)$ variables in each iteration, and that m variables are selected, the overall time complexity of the forward selection with LOO criterion is $\mathcal{O}(m^2n^3d)$.*

Remark 3.2 (Complexity using the efficient LOO implementation)**.** *If the efficient LOO computation is instead used, the overall complexity reduces to $\mathcal{O}(m^2n^2d)$.*

In the work by [40], the SVM is solved only once with the full set of variables. Although the selection criterion used is the change on the norm weight cost function, the support vectors $\boldsymbol{\alpha}$ remain fixed throughout the whole search. Such strategy reduces the complexity of such approach from n-QP problems to just one. Although efficient, such approach violates the optimality conditions and relies in a bound on the generalization performance. In the next sections, the use of low rank modifications is motivated as to exploit the structure of the LS-SVM classifiers. We will show how the computational complexity can be still reduced to $\mathcal{O}(mn^2d)$, thus alleviating the computational load of wrapper algorithms for a large number of variables.

3.4 Low rank modifications

In many applications it is often desired to solve a sequence of modified least squares problems where in each step rows of data are added, deleted, or both. This need arises, for instance, when data are arriving sequentially. In other applications columns of the data matrix may be added or deleted. Such modifications are usually referred to as updating or downdating the least squares solutions. Important applications where modified least squares problems arise include statistics, optimization, and signal processing.

In the case that the inverse of a general matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is known, it may also be useful to know how the inverse changes upon addition of a matrix of small rank [39]. This section motivates that in this particular situation, it is much more efficient to *update* the inverse of $\mathbf{A}$ than to generate it again from *scratch*. We are interested in the generic system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A}$ is $n \times n$ matrix and $\mathbf{b}$ is a n-vector. Once $\mathbf{x}$ has been computed, it is often necessary to solve the modified system $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$. Clearly, we should be

able to modify $\mathbf{A}$ to obtain factors for $\tilde{\mathbf{A}}$, from which $\tilde{\mathbf{x}}$ may be computed as before. In this section, special attention is devoted to one type of modification, in which $\tilde{\mathbf{A}}$ has the form $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{u}\mathbf{u}^\top$, where $\mathbf{u} \in mathbbR^n$. The matrix $\mathbf{u}\mathbf{u}^\top$ is a matrix of *rank-one* [39], and the problem is usually described as that of *updating* the factors of $\mathbf{A}$ following a *rank-one* modification. Such type of modifications have been extensively studied in [37].

There exist mainly two methods that efficiently solve this problem. The first approach is based on the Sherman-Morrison-Woodbury (SMW) formula [39] for a low rank update of the inverse of a matrix. The second approach is based on the *Cholesky* factorization of the matrix $\mathbf{A}$. The Sherman-Morrison-Woodbury formula has been used widely in the context of interior point methods (IPM) for linear programming (LP). In that context, however, the method may run into numerical difficulties. The Cholesky factorization is somewhat more expensive than the SMW method, with two times the workload and about three times storage requirement of the SMW method [27, 93]. On the other hand, the Cholesky method is numerically more stable [39]. In SVM, the Cholesky factorization has been previously used to efficiently solve the IPM [32] and for incremental training [96]. Also in [93], the use of Cholesky decompositions is motivated for low rank modifications.

3.4.1 SMW formula and rank-one modifications

The SMW formula gives the inverse of a matrix modified by an arbitrary rank. Let $\mathbf{H} \in \mathbb{R}^{n\times n}$ be a square invertible matrix, whereas $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{n\times q}$ are two matrices with $q \leq n$. Assume also that $\mathbf{R} = \mathbf{I}_q + \beta\mathbf{V}^\top\mathbf{H}^{-1}\mathbf{U}$, $\mathbf{R} \in \mathbb{R}^{q\times q}$ is invertible, where $\mathbf{I}_q \in \mathbb{R}^{q\times q}$ denotes the identity matrix and $\beta \in \mathbb{R}$ a arbitrary scalar. Then the following holds [118]

$$(\mathbf{H} + \beta\mathbf{U}\mathbf{V}^\top)^{-1} = \mathbf{H}^\top - \beta\mathbf{H}^{-1}\mathbf{U}\mathbf{R}^{-1}\mathbf{V}^\top\mathbf{H}^{-1} \ . \tag{3.10}$$

The motivation for this formula resides in the fact that if q is much smaller than n, then $\mathbf{R}$ is easier to invert than $\mathbf{H} + \beta\mathbf{U}\mathbf{V}^\top$. If the modification to $\mathbf{H}$ has rank $q = 1$, then $\mathbf{U} = \mathbf{u} \in \mathbb{R}^n$ and $\mathbf{V} = \mathbf{v} \in \mathbb{R}^n$, in which case $\mathbf{R} = \rho$ is a scalar, $\rho = 1 + \beta\mathbf{v}^\top\mathbf{H}^{-1}\mathbf{u} \neq 0$. Hence (3.10) becomes [94]

$$\left(\mathbf{H} + \beta\mathbf{u}\mathbf{v}^\top\right)^{-1} = \mathbf{H}^{-1} - \frac{\beta}{\rho}\mathbf{H}^{-1}\mathbf{u}\mathbf{v}^\top\mathbf{H}^{-1} \ , \tag{3.11}$$

Since any rank-one correction to $\mathbf{H}$ can be written as $\beta\mathbf{u}\mathbf{v}\top$, then (3.11) gives the rank-one change to its inverse. The proof is by direct multiplication. Setting $\beta = +1$ corresponds to an *rank-one update* to $\mathbf{H}$, whereas $\beta = -1$ refers to a *rank-one downdate*. In the case of *symmetric* modifications, we distinguish between $\mathbf{H} + \mathbf{u}\mathbf{u}^\top$ (positive rank-1, update) and $\mathbf{H} - \mathbf{u}\mathbf{u}^\top$ (negative rank-1, downdate). These two instances are fundamentally different in that, for a positive update the modified matrix remains positive definite if the initial $\mathbf{H}$ is positive definite. On the other hand, negative downdates may break down and can result in large errors if $\mathbf{H}$ is close to singularity [27, 39]. Low rank updates of the form $\tilde{\mathbf{H}} = \mathbf{H} \pm \mathbf{U}\mathbf{U}^\top$, $\mathbf{U} \in \mathbb{R}^{n\times q}$, can be performed by applying q updates/downdates sequentially [93]. For practical computation of (3.11), one solves the linear systems $\mathbf{H}\mathbf{a} = \mathbf{u}$ and $\mathbf{H}\mathbf{b} = \mathbf{v}$ for $\mathbf{a}$ and $\mathbf{b}$, using the known $\mathbf{H}^{-1}$. Compute $\rho = 1 + \beta\mathbf{v}^\top\mathbf{a}$. If $\rho \neq 0$, the change to $\mathbf{H}^{-1}$ is $-(\beta/\rho)\mathbf{a}\mathbf{b}^\top$.

3.4.2 Cholesky factorization

Let us consider again a general matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, being symmetric and positive definite. Then, for $\mathbf{A}$ there exists a lower triangular matrix $\mathbf{L}$ with positive diagonal elements such that $\mathbf{A} = \mathbf{L}\mathbf{L}^\top$. The matrix $\mathbf{L}$ is unique and is known as the Cholesky factor of $\mathbf{A}$ [39]. Several computations involving $\mathbf{A}$ can be done using $\mathbf{L}$. For instance, the system $\mathbf{A}\mathbf{x} = \mathbf{b}$, can be solved by means of two triangular systems as $\mathbf{L}\mathbf{t} = \mathbf{b}$, $\mathbf{L}^\top\mathbf{x} = \mathbf{t}$.

Suppose now that we use a representation based on a Cholesky decomposition for the matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$ as $\mathbf{H} = \mathbf{L}\mathbf{L}^\top$, and we further consider the case where $\mathbf{H}$ is modified by a symmetric matrix of rank-one, i.e., we have $\tilde{\mathbf{H}} = \mathbf{H} + \mathbf{u}\mathbf{u}^\top$. Assuming that the Cholesky factors of $\mathbf{H}$ are known, we are now interested in the factors of

$$\tilde{\mathbf{H}} = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top \quad . \tag{3.12}$$

The method that calculates $\tilde{\mathbf{L}}$ from $\mathbf{L}$ and $\mathbf{u}$ is called a *rank-one* update algorithm. One way of performing such update is by applying a series of orthogonal *Givens* rotations of the form

$$\mathbf{G}_{n-1}\ldots\mathbf{G}_k\ldots\mathbf{G}_1\left[\mathbf{u},\mathbf{L}\right]^\top = \left[\mathbf{0},\tilde{\mathbf{L}}\right]^\top \quad , \tag{3.13}$$

leading to $\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top = \mathbf{L}\mathbf{L}^\top + \mathbf{u}\mathbf{u}^\top$. The algorithm proceeds $k = 1, \ldots, n-1$. The matrix $\mathbf{G}_k$ is chosen based on the k-th column, namely $\mathbf{G}_k[\mathbf{u}_k, \mathbf{L}_{kk}]^\top =$

$[0, \tilde{\mathbf{L}}_{kk}]^\top$. Appropriate algorithms can be found in standard texts [37, 39] and software packages such as the LAPACK library [12]. Additionally, the routine `cholupdate` from MATLAB © [61] also performs such operation.

In the next section we will concentrate our attention to the use of the SMW method and its application to the LS-SVM classifiers. The reason behind this choice is related to the algorithm's simplicity to update the LS-SVM classifier. Nevertheless, the Cholesky update counterparts are equally applicable to the LS-SVM classifier and with respect to a rank-one modification.

3.5 Low rank updated LS-SVM

In this section, we describe how to perform low rank modifications to the system of linear equations (3.1) for the LS-SVM classifier. In fact, we refer to modifications involving $\mathbf{H}$ or its corresponding inverse $\mathbf{H}^{-1}$. To ensure positive-definiteness of $\mathbf{H}$, we rely on an adequate selection of the hyper-parameter γ.

Recalling the basic mechanism of *wrapper* search methods described earlier on section 3.1, it is clear that a repetitive evaluation of the learning machine would be required each time a new variable is tested. For instance, in the LS-SVM classifier, the calculation of $\mathbf{H}^{-1}$ for each candidate would be demanded. Such alternative becomes prohibitive for large data sets with high number of variables. However, there exists a way to circumvent such repetitive matrix inversions in the case of a *linear kernel*.

Consider again the training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$ which is described by the set of variables $\mathcal{S} = \{\mathbf{x}^k\}_{k=1}^d, \mathbf{x}^k \in \mathbb{R}^{n\times 1}$, and given that $\mathbf{H}^{-1}$ is known and positive definite, then for a linear kernel we can write

$$\begin{aligned}\mathbf{H} &= \boldsymbol{\Omega} + \gamma^{-1}\mathbf{I}_n \\ &= \sum_{k=1}^{d} \mathbf{x}^k \mathbf{x}^{k\top} + \gamma^{-1}\mathbf{I}_n \ .\end{aligned} \tag{3.14}$$

Splitting the sum at the level of variable k, and grouping back the first two terms, we have

$$\mathbf{H}_k = \sum_{j=1}^{k-1} \mathbf{x}^j {\mathbf{x}^j}^\top + \gamma^{-1}\mathbf{I}_n + \mathbf{x}^k {\mathbf{x}^k}^\top$$

$$\mathbf{H}_k = \mathbf{H}_{k-1} + \mathbf{x}^k {\mathbf{x}^k}^\top \quad . \tag{3.15}$$

Direct application of (3.11) to (3.15) results in

$$\mathbf{H}_k^{-1} = \mathbf{H}_{k-1}^{-1} - \frac{\mathbf{H}_{k-1}^{-1}\mathbf{x}^k {\mathbf{x}^k}^\top \mathbf{H}_{k-1}^{-1}}{1 + {\mathbf{x}^k}^\top \mathbf{H}_{k-1}^{-1}\mathbf{x}^k} \quad . \tag{3.16}$$

The previous set of equations explain us that, for a linear kernel and a given variable $\mathbf{u}_k$ to be tested for inclusion, the inverse of $\mathbf{H}^{-1}$ can be updated from the inverse of the previous step. Combination of this rank-one update procedure with the LOO error given in section 3.3, provides an efficient algorithm for variable selection. With the use of such *low rank* modifications, algorithms for *forward* and *backward* search can be devised by direct application of *updates* and *downdates* operations respectively. There is no computation cost incurred other than matrix vector multiplications. Moreover, the updated matrix at each step, can be immediately used to compute the efficient LOO error earlier outlined in Algorithm 1.

3.5.1 Forward update

In forward selection, we start from the empty set of variables. In the case of LS-SVM this translates into $\mathbf{H}_0 = \gamma^{-1}\mathbf{I}_n$, and so the initial inverse being $\mathbf{H}_0^{-1} = \gamma\mathbf{I}_n$. Therefore, we keep updating this matrix when testing the variable $\mathbf{x}^k$ for inclusion. Notice that no matrix inversions are required at all since the by-product of the rank-one update is directly used to compute the LOO error. In short, the goal is to compute $\mathbf{H}_k^{-1}$ at selection step k from $\mathbf{H}_{k-1}^{-1}$ at step $k-1$, upon addition of variable x^k

$$\mathbf{H}_k^{-1} = \mathbf{H}_{k-1}^{-1} - \frac{\mathbf{H}_{k-1}^{-1}\mathbf{x}^k {\mathbf{x}^k}^\top \mathbf{H}_{k-1}^{-1}}{1 + {\mathbf{x}^k}^\top \mathbf{H}_{k-1}^{-1}\mathbf{x}^k} \quad . \tag{3.17}$$

3.5.2 Backward downdate

Along the same lines of the *forward update* algorithm and using the elements we have described in the previous sections, we can come up with an efficient algorithm for backward elimination of variables. Since we start from the full set of all variables, the inverse of the matrix $\mathbf{H}$ is required only once to initialize the algorithm. Afterwards, at every elimination step the inverse $\mathbf{H}^{-1}$, is *downdated*. The basic idea is to compute $\mathbf{H}_k^{-1}$ at elimination step k, from $\mathbf{H}_{k+1}^{-1}$ at step $k+1$, upon removal of variable x^k, that is

$$\mathbf{H}_k^{-1} = \mathbf{H}_{k+1}^{-1} + \frac{\mathbf{H}_{k+1}^{-1}\mathbf{x}^k{\mathbf{x}^k}^\top\mathbf{H}_{k+1}^{-1}}{1-{\mathbf{x}^k}^\top\mathbf{H}_{k+1}^{-1}\mathbf{x}^k} \quad . \tag{3.18}$$

Since the formulations given in this section refer to the *linear kernel*, the rank-one modifications expressions are restricted to this case. The use of a non-linear kernel strictly requires the computation of $\mathbf{H}^{-1}$ at each selection step. The algorithm for forward updates is summarized in Algorithm 2. The backward downdate version possesses a similar structure and is given in Algorithm 3.

Remark 3.3 (Complexity of the proposed algorithm)**.** *To select m variables, the outermost loop requires m runs. The middle loop evaluates in turn the LOO for each of the remaining c variables. Using equations (3.9)-(3.7), the LOO predictions demand only $\mathcal{O}(n)$ operations. Thereby, the complexity is dominated by $\mathcal{O}(n^2)$ matrix-vector operations to update $\mathbf{H}^{-1}$ in line 8. Thus, the overall complexity is $\mathcal{O}(mn^2d)$.*

Similar efficient derivations for the PRESS statistic have been previously described in the literature. For instance the authors in [46], derive a recursive computation formula for PRESS errors in the regularized orthogonal least squares (OrLS) and forward regression. The PRESS statistic is then based on an orthogonal model with a diagonal Hessian matrix. There, the matrix inversion lemma avoids matrix inversions and further reduces computations. Nevertheless, these parametric models depend on the number of variables d, and hence the design matrix and weight vector $\mathbf{w} \in \mathbb{R}^d$ grow as more variables enter model. In contrast, the dual formulation of the PRESS statistic in (3.7) is independent of the number of variables, since both the model parameters $\boldsymbol{\alpha} \in \mathbb{R}^n$ and matrix $\mathbf{H}$ scale with the number of data points n. Hence, working in a primal-dual offers wider modeling flexibility (cf. Remark 2.3). Further improvements to our

methods have been recently proposed in [78], where primal-dual formulations are alternatively used to yield algorithms that scale linearly with n. In the following section, we present results of these two algorithms when applied on a series of benchmark data sets provided in [87]. The computational efficiency is compared with respect to existing approaches for variable selection.

Algorithm 2 Algorithm for forward variable selection. We apply *rank-one* updates into the linear kernel matrix, to obtain a sequential forward selection (SFS) algorithm. Such modification avoids completely the use of matrix inversion type of operations.

Require: $\mathcal{D} = \{(\mathbf{x}_i, y_i\}_{i=1}^n$: training set; n: number of data points; d: number of variables.; m: number of variables to select.

Ensure: Selected variables $S \neq \emptyset$.

$S = \emptyset$ {final set of selected variables}
$P = \{\mathbf{x}^1, \ldots, \mathbf{x}^k, \ldots, \mathbf{x}^d\},\ k = 1, \ldots, d.$ {full set of variables}
$\mathbf{H}^{-1} \leftarrow \gamma \times \mathbf{I}_n$ {Initial inverse when no variables}
for $k = 1$ to m **do**
  $loo \leftarrow \infty$
  $c \leftarrow \mathrm{card}\{P\}$ {Variables yet to evaluate}
  for $l = 1$ to c **do**
    $\mathbf{H}_l^{-1} \leftarrow \mathbf{H}^{-1} - (\mathbf{H}^{-1}\mathbf{x}^l{\mathbf{x}^l}^\top\mathbf{H}^{-1})/(1+{\mathbf{x}^l}^\top\mathbf{H}^{-1}\mathbf{x}^l)$ {Update inverse}
    Solve for b_l , $\boldsymbol{\alpha}_l$ using $\mathbf{H}_l^{-1}$ and (3.4). {Solve LS-SVM system}
    Compute loo_l using first (3.9), then (3.7) and (2.28). {Efficient LOO error}
    if $loo_l < loo$ **then**
      {Save the loo_l, index variable l and $\mathbf{H}_l^{-1}$ for next comparison}
      $loo \leftarrow loo_l$
      $l^* \leftarrow l$
      $\mathbf{H}_*^{-1} \leftarrow \mathbf{H}_l^{-1}$
    end if
  end for
  $\mathbf{H}^{-1} \leftarrow \mathbf{H}_*^{-1}$ {Store inverse for next selection procedure}
  $S \leftarrow S \cup \mathbf{x}^{l^*}$ {Update selected variables}
  $P \leftarrow P \setminus \mathbf{x}^{l^*}$ {Remove selected variable from the full set}
end for
return S

Algorithm 3 Algorithm for sequential backward elimination. We apply *rank-one* downdates into the linear kernel matrix, to obtain a fast sequential backward elimination (SBE) algorithm. In this algorithm, we require to compute the inverse of the matrix $\mathbf{H}$ just one time. Downdates are performed iteratively on this matrix.

Require: $\mathcal{D} = \{(\mathbf{x}_i, y_i\}_{i=1}^n$: training set; n: number of data points; d: number of variables.

Ensure: $S \neq \emptyset$: selected variables.

$S = \{\mathbf{x}^1, \ldots, \mathbf{x}^k, \ldots, \mathbf{x}^d\},\ k = 1, \ldots, d.$ {full set of candidates}
$P = \emptyset$. {variables to remove}
Compute $\mathbf{H}^{-1} = (\boldsymbol{\Omega} + \gamma^{-1}\mathbf{I}_n)^{-1}$ with all variables. {Initial inverse}
while $S \neq \emptyset$ **do**
  $c \leftarrow \mathrm{card}\{S\}$ {Variables yet to evaluate}
  for $l = 1$ to c **do**
    $\mathbf{H}_l^{-1} \leftarrow \mathbf{H}^{-1} + (\mathbf{H}^{-1}\mathbf{x}^l{\mathbf{x}^l}^\top\mathbf{H}^{-1})/(1 - {\mathbf{x}^l}^\top\mathbf{H}^{-1}\mathbf{x}^l)$ {Downdate inverse}
    Solve for b_l , $\boldsymbol{\alpha}_l$ using $\mathbf{H}_l^{-1}$ and (3.4). {Solve LS-SVM system}
    Compute loo_l using first (3.9), then (3.7) and (2.28). {Efficient LOO error}
  end for
  $loo^* \leftarrow \min loo_l, l = 1, \ldots, c$ {Lowest LOO error}
  $\mathbf{x}^{l^*} \leftarrow \arg\min loo_l,\ l = 1, \ldots, c$ {Variable to be definitely removed}
  $\mathbf{H}^{-1} \leftarrow \mathbf{H}^{-1} + (\mathbf{H}^{-1}\mathbf{x}^{l^*}{\mathbf{x}^{l^*}}^\top\mathbf{H}^{-1})/(1 - {\mathbf{x}^{l^*}}^\top\mathbf{H}^{-1}\mathbf{x}^{l^*})$ {Downdate inverse with respect to $\mathbf{x}^{l^*}$}
  $S \leftarrow S \setminus \mathbf{x}^{l^*}$ {Remove selected variable from the full set}
  $P \leftarrow P \cup \mathbf{x}^{l^*}$ {Add removed variable to set of removed variables}
end while
return S

3.6 Empirical Results

In this section, experimental results demonstrate the benefits on the use of low rank updates for LS-SVM classifiers. This approach overcomes, and reduces, the high computational cost incurred during variable selection. Table 3.6 shows a selection of seven public domain benchmark data sets used in [87]. For six out

of seven data sets, one hundred random partitions of training and test sets are available. The SPLICE data set contains only twenty. Linear and RBF kernels are considered in combination with the two search algorithms: forward and backward search. The use of low rank updates/downdates is only applicable for the case of linear kernel. The selection criterion is based on the PRESS statistic derived from the efficient LOO estimator. Additionally, results in two public available microarray data are also included.

Data set	Training data	Test data	Number of randomizations	Input variables
BREAST CANCER	200	77	100	9
DIABETES	468	300	100	8
FLARE SOLAR	666	400	100	9
GERMAN	700	300	100	20
HEART	170	100	100	13
SPLICE	1000	2175	20	60
WAVEFORM	400	4600	100	21

Table 3.1: Data sets used in benchmark study.

3.6.1 Computational performance

First, we compare the computational cost incurred when using either of the two proposed algorithms on the same data set. Algorithms 2 and 3 are tested on a balanced two-class synthetic data set with five hundred data points and one thousand variables, that is $\{n = 500, d = 1000\}$. The main computational difference between both approaches is the *initial inverse matrix* $\mathbf{H}_0^{-1}$. Therefore, this $n \times n$, $n = 500$, kernel matrix needs to be first computed and then inverted in the *rank-one downdates* algorithm (Refer to line 3 in Algorithm 3). Clearly, such situation harms the performance of the rank-one *downdate* algorithm over its *update* counterpart. Figure 3.1 shows the computational time for both algorithms with respect to the number of variables. The gap between both algorithms clearly appears in the very first iteration, which represents the effect of the initialization scheme.

Additionally, to assess the computational complexity of the variable selection problem and to show the improvements especially in the use of low rank updates,

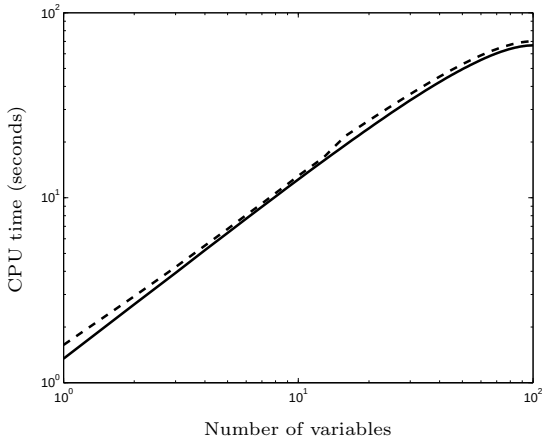


Figure 3.1: Computational time for the *update* (Solid line) and *downdate* (Dashed line) algorithms on a balanced two-class synthetic data set with $n = 500$ data points and $d = 1000$ variables. The main computational difference between both approaches is the *initial inverse matrix* $\mathbf{H}_0^{-1}$. The gap between both algorithms clearly appears in the very first iteration, which represents the effect of the initialization scheme. In the *downdate* algorithm we require first to compute the matrix $\mathbf{H} = \mathbf{\Omega} + \mathbf{I}_n$, and its associated inverse; whereas in the *update* algorithm we start simply from the diagonal matrix $\mathbf{H} = \gamma^{-1}\mathbf{I}_n$.

we consider two experiments. First, the SPLICE data set is used to see the impact of having a moderate number of data points (1000 training) and variables (60 variables). The second experiment constitutes the opposite case, that is a very small number of data points and a large number of variables. For this purpose the Colon cancer data set [7] is employed. It is clear that the use of a wrapper method pose difficulties in both the number of data points and the number of variables of the data set under consideration. The computational performance on each of the examples mentioned is shown in Figure 3.2 and Figure 3.3. In the SPLICE data set we want to see the effect of inverting a reasonable big matrix, therefore we consider a naive implementation where all the required matrix inversions are performed. The inversion at the first iteration already pushes up the computational load. A dramatic reduction is immediately appreciated when using the low rank updated algorithm even for selecting a small number of variables. In the Colon cancer data set, the efficiency of the rank-update algorithm, is compared with other two variants that also employ the LS-SVM

classifier. The first algorithm presented in [102], uses a closed-form to compute the LOO error. Matrix inversions are, however, still required to solve the LS-SVM system for each new potential subset of variables. The second approach, given in [122], makes use of a span-based [112] upper bound on the LOO error. The expensive part of the algorithm in [122] is mainly the computation of a $n \times n$ distance matrix each time a new subset of variables is tested. We stop the three algorithms when 200 out of the 2000 variables have been selected in the Colon cancer data set and compare the computational time. Clearly, it can be drawn that the proposed algorithm scales very nicely even for high dimensional data sets. The time gap with respect to the approach in [102] is readily appreciable (around two orders of magnitude). When compared to the bound approach [122], our proposed method is still faster, despite we compute an exact form of LOO error.

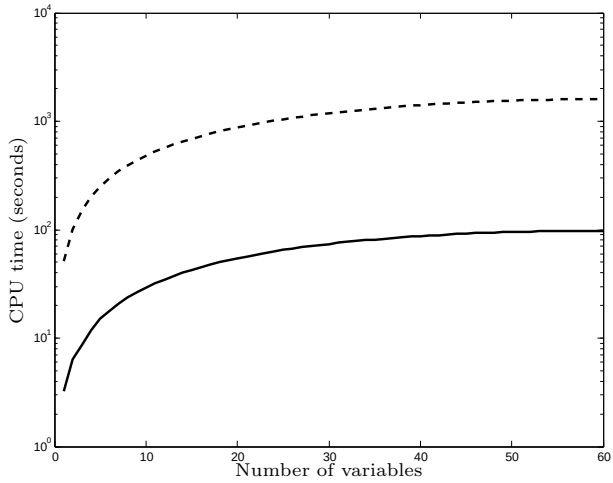


Figure 3.2: Computational time of the low *rank update* (Solid line) algorithm when compared to a naive implementation (Dashed line) where all matrix inversions are performed. The simulations are done with the SPLICE data set. Although the number of variables is 60, we can immediately see a dramatic reduction in the time needed to select even a small number of variables.

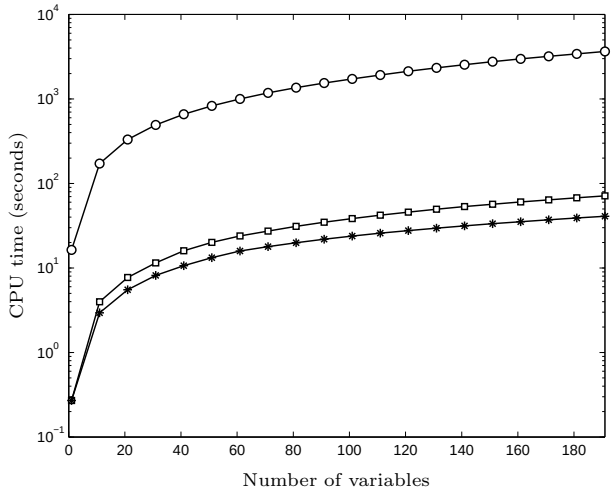


Figure 3.3: Computational time for the low *rank update* algorithm ($*$) when compared to other two variable selection algorithms also based on the LS-SVM classifier. The first algorithm ($\circ$) by [102] requires of matrix inversions during the variable selection phase. The second variant ($\square$) proposed in [122] makes use of an upper bound on the LOO error but requires of an $n \times n$ distance matrix to be computed at every selection step. We test the three algorithms in the Colon cancer data set and show their performance to select 200 out of a total of 2000 variables. The gap with respect to the approach in [102] is readily appreciable (around two orders of magnitude). When compared to the bound approach [122], our proposed method is still faster, despite we compute an exact form of LOO error.

Likewise, the *downdate* algorithm is applied to the Colon data and it is compared to two variants of the SVM-RFE algorithm [40]. The first variant (SVM-RFE1), computes the support vectors at each step of the elimination phase (retraining), whereas in the second variant (SVM-RFE2) the support vectors remain fixed from the first iteration (no retraining). Figure 3.4 shows the computational time required in the elimination of 500 variables out of 2000. The QP problem involved in the standard SVM, as mentioned earlier in section 3.2, is more computationally expensive than the LS-SVM solution. The sequential nature of the backward search increases the volume of computations. The proposed backward approach is even faster than the SVM-RFE2, which avoids a considerable number of operations by keeping support vectors fixed throughout

the whole elimination phase.

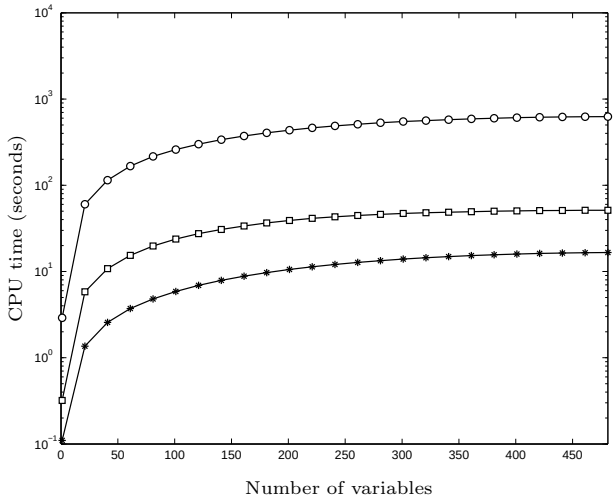


Figure 3.4: Computational time for the low *rank downdate* algorithm ($*$) when compared with two SVM-RFE ([40]) based algorithms. The first variant, SVM-RFE1 ($\circ$), computes the support vectors at each step of the elimination phase (retraining), whereas in the second variant,SVM-RFE2 ($\square$), support vectors are fixed as from the first iteration (no retraining). We stop the algorithm when 500 variables of a total of 2000 are removed in the Colon cancer data set. Our proposed approach is even faster than the SVM-RFE2, which avoids a considerable number of operations by keeping fixed the support vectors through the whole elimination phase.

Clearly, the combination of the low rank updates and the closed-form expression of the LOO error benefits the LS-SVM classifiers for the task of variable selection. It is important to emphasize that low rank update modifications for the linear kernel does not require matrix inversions. Instead it simply uses matrix-vector operations. Similar results are obtained for the linear kernel in combination with the backward downdate.

3.6.2 Benchmarking data

The final performance of any of the possible strategies (and combinations) is heavily subjected to the hyperparameter selection, i.e. a correct choice for

$\boldsymbol{\theta} = (\gamma, \sigma)$ when using an RBF kernel or simply γ for the linear kernel. In this work, the hyperparameters are tuned as to minimize the PRESS statistic. For each of the 100 realizations of the data sets (20 for the SPLICE data set), we select the pair (γ^*, σ^*) minimizing the LOO PRESS statistic. Such pair is then used during the variable selection procedure. Tables 3.2 and 3.3 show the performance of the LS-SVM with respect to the error rate over the corresponding number of realization per data set. The basic LS-SVM classifiers with no variable selection, but appropriate hyperparameter tuning, constitutes the baseline classifier. Linear kernel and RBF kernel are used.

Data set	LS-SVM	LS-SVM+SFS	LS-SVM+SBE
BREAST CANCER	0.268 (0.044)	0.268 (0.052) {2}	0.264 (0.049) {2}
DIABETES	0.221 (0.013)	0.234 (0.017) {6}	0.234 (0.017) {6}
FLARE SOLAR	0.334 (0.014)	0.329 (0.018) {2}	0.326 (0.019) {2}
GERMAN	0.252 (0.020)	0.245 (0.021)	0.242 (0.022) {16}
HEART	0.151 (0.026)	0.157 (0.032)	0.157 (0.032)
SPLICE	0.161 (0.007)	0.161 (0.006) {27}	0.161 (0.007) {28}
WAVEFORM	0.133 (0.006)	0.132 (0.006)	0.147 (0.009)

Table 3.2: Error rates of LS-SVM classifier with a linear kernel. We apply *rank-one updates* into the linear kernel matrix to perform fast sequential forward selection (SFS) of variables. Similarly *rank-one downdates* into the linear kernel matrix allows for fast sequential backward elimination (SBE) of variables. The results for each method correspond to the mean error rate over test data for 100 realizations of each data set (20 for the splice data set). Whenever a lower number of variables can be selected without degrading the performance of the classifier, this number is indicated between curly brackets .

In general terms, there seems to be little difference in the performances regardless the search methodology chosen. Both forward and backward search methods provide very similar results across all data sets when combined either with the linear kernel or the RBF kernel. The number of selected variables tends to be the same number as well. The computational cost, however, when combined with linear kernels is significantly several orders of magnitude lower thanks to the *low rank* modifications introduced. Additionally, in three of the data sets, namely: BREAST, DIABETES and FLARE SOLAR, there is apparently no gain in performance with respect to the baseline LS-SVM. There is, however,

Data set	LS-SVM	LS-SVM+SFS	LS-SVM+SBE
BREAST CANCER	0.258 (0.064)	0.250 (0.047) {2}	0.258 (0.047) {2}
DIABETES	0.235 (0.015)	0.233 (0.019) {7}	0.233 (0.019) {7}
FLARE SOLAR	0.344 (0.011)	0.326 (0.019) {2}	0.326 (0.019) {2}
GERMAN	0.232 (0.025)	0.235 (0.022)	0.235 (0.022)
HEART	0.155 (0.034)	0.167 (0.034)	0.167 (0.034)
SPLICE	0.107 (0.007)	0.075 (0.006) {15}	0.077 (0.006) {14}
WAVEFORM	0.096 (0.004)	0.099 (0.005) {16}	0.098 (0.004)

Table 3.3: Error rates of LS-SVM classifier with RBF kernel. The purpose of this experiment is for completeness. Although we can not apply low rank modifications as in the case of the linear kernel, it is interesting to see how does the RBF kernel behaves in variable selection problems. To implement sequential forward selection (SFS) and sequential backward elimination (SBE) with the RBF kernel we do require of matrix inversions at every stage. The results for each method correspond to the mean error rate over test data for 100 realizations of each data set (20 for the splice data set). Whenever a lower number of variables can be selected without degrading the performance of the classifier, this number is indicated between curly brackets.

a consistent tendency of the other LS-SVM variants to select a lower number of variables with no substantial increase of the test error. Since the number of parameters is relatively small in all of these three data sets and the number of data points is not that large, the search space appears to be well covered. We show in Figure 3.5 the test error on the FLARE data set. Boxplots are generated from the available 100 realizations, explaining an almost constant variance remains with median value clearly low at two variables. The low rank downdate algorithm was used.

On the other hand, for the SPLICE data set, the RBF kernel performs significantly better than any other alternative. Additionally, the RBF kernel tends to select less variables. Such an improvement, however, is attained at a much higher computational cost mainly because of the large number of data points (n = 1000). The low ranked LS-SVM with linear kernel is obviously faster than the RBF kernel setting, but produces slightly higher error when using the top-selected 28 variables. Indeed, Figure 3.6 explains, in terms of the test error, our previous comments on the SPLICE data set.

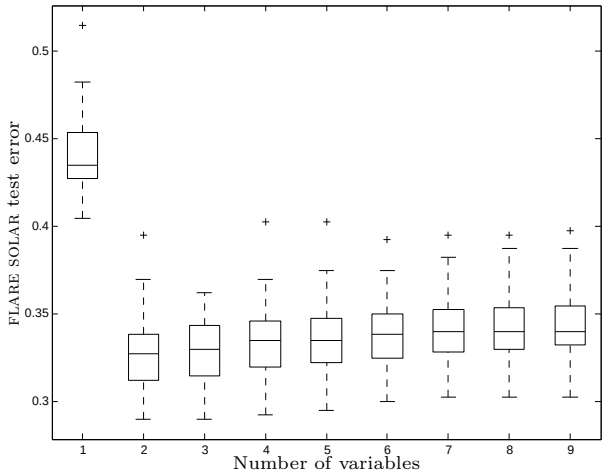


Figure 3.5: FLARE data set. Test error boxplots averaged over 100 realizations. The low rank downdate algorithm was used.

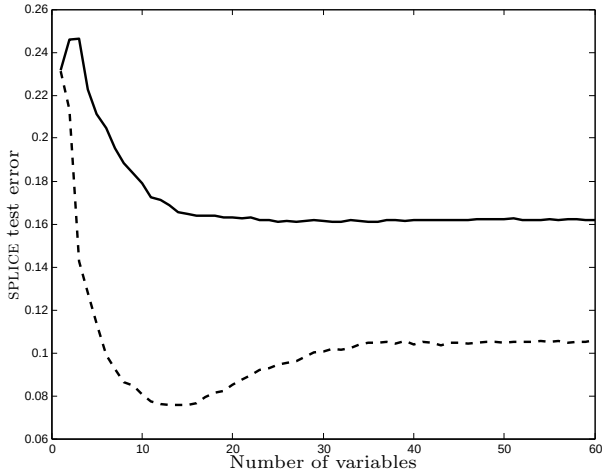


Figure 3.6: SPLICE data set test error averaged over 20 realizations. (*Solid*) Linear kernel with low rank downdates. (*Dashed*) RBF kernel with explicit matrix inversion.

Another interesting example is the HEART data set. This data set tends to produce the same results regardless of the method used. In fact the basic LS-SVM, with no variable selection, appears more consistent for this data set. A plausible explanation could be the small size of this data set, thus a high risk of overfitting is present.

3.6.3 Microarray data

To assess the performance in classification, the full data set is partitioned into (nearly balanced) training and test sets. An external 10-fold cross-validation, as mentioned in [10], is applied to the training set. On each fold the variable selection is done using the LOO error. This methodology aims at reducing the selection bias given the low number of data points. Variables consistently selected in different folds comprise the set used to compute the test error.

Figure 3.7, shows the test error (with respect to the number of selected variables) of the proposed low rank update algorithm when applied to the Colon data set. Additionally, we make comparisons with respect to the methods for variable selection cited earlier in [122] and [102]. Our proposed algorithm exhibits better stability than the bound [122] approach. The tightness of such bound is heavily Dependant on the stability of the LS-SVM to perturbations in the training data. The exact computation of the LOO has therefore a direct impact on the test error. The variance of the cross-validation is depicted as error bars. Differences are less remarkable as the number of variable increases.

The low rank downdate algorithm is compared using the Leukemia cancer data set. We again use the two SVM-RFE algorithms [40]. The test error for 1000 out of 2000 selected variables is shown in Figure 3.8. The use of the LOO error is clearly superior to the margin criterion used in the SVM-RFE algorithm for eliminating variables. This was expected since the margin is a bound on the generalization error. In addition, the test error is remarkably lower around 200 variables. The latter observation suggest of a more compact and accurate representation of the Leukemia data.

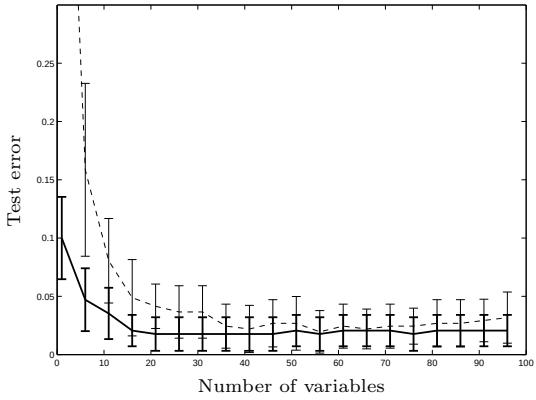


Figure 3.7: Test error on the Colon data set. We compare the performance of our low rank update algorithm (Solid) with the algorithm by [122] (Dashed). Error bars shows the variance of the external 10-fold cross-validation. We stop both algorithms when 100 variables out of 2000 were selected.

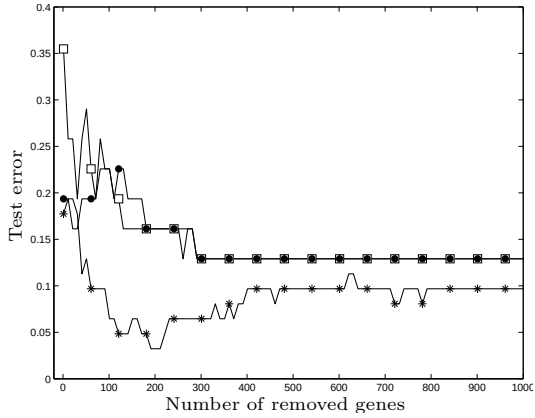


Figure 3.8: Test error on the Leukemia data set. We compare the low *rank downdate* algorithm ($*$) with the two SVM-RFE ([40]) based algorithms: SVM-RFE1 ($\bullet$) and SVM-RFE2 ($\square$). Test error is shown when 1000 variables (out of 2000) have been removed. The use of the LOO error is clearly superior to the margin criterion used in the SVM-RFE algorithm for eliminating variables.

3.6.4 Model selection

In the experiments reported in earlier sections, we tuned the hyperparameters with the full set of variables and leave them unmodified during the variable selection process. The main reason for this was, in principle, to avoid more degrees of freedom within the variable selection loop, as well as to reduce the computational load. However, performing a careful adjustment of the hyperparameters, once a variable is definitely added or removed to the final set, may result in a more robust and consistent set of variables. Instead of considering a grid search approach, we chose for a direct optimization of the PRESS statistic with respect to the LS-SVM parameters $\boldsymbol{\theta} = (\gamma, \sigma)$ (or simply γ for a linear kernel). Since the PRESS statistic is a continuous function, we can make use of standard simplex-search algorithm, such as the MATLAB routine `fminunc`. An adequate practice when using unconstrained optimization routines, is that of transforming the hyperparameters into a logarithmic scale. Another alternative to ensure correct values for the parameters, e.g. $\gamma > 0$, is the `fmincon` function with the additional non-negative constraint for $\boldsymbol{\theta}$. Additionally, extreme values for the hyperparameters may introduce instabilities and singularities into the kernel matrix, and consequently numerical problems in the low rank updates/downdates. Alternatively to the use of the PRESS statistic, the authors in [21] propose to add a regularization/penalization term to prevent overfitting and perform a subsequent optimization within a Bayesian framework. Instead, we simply add a second term to the PRESS statistic that accounts for the norm of the hyperparameters

$$\text{error}_{\text{pressreg}} = \frac{1}{n}\sum_{i=1}^{n}\left(r_i^{(-i)}\right)^2 + \frac{1}{2}\sum_{j=1}^{n_{\boldsymbol{\theta}}}\theta_j^2 \ , \tag{3.19}$$

where $n_{\boldsymbol{\theta}}$ denotes the number of parameters. We test the modified PRESS criterion in the HEART data set. The reduced number of training data points of the HEART data set supports it as a good example. In Figure 3.9 and 3.10, the test error box plots for the HEART data with the PRESS statistic and the modified version are shown respectively. Both results were obtained using the LS-SVM with RBF kernel and the forward search. Although the true error seems to increase as we add more and more variables, the modified PRESS estimator did capture the trend of the true error. The sharp point, in Figure 3.10, of the error is a clear indicator of a good candidate model with less number of variables.

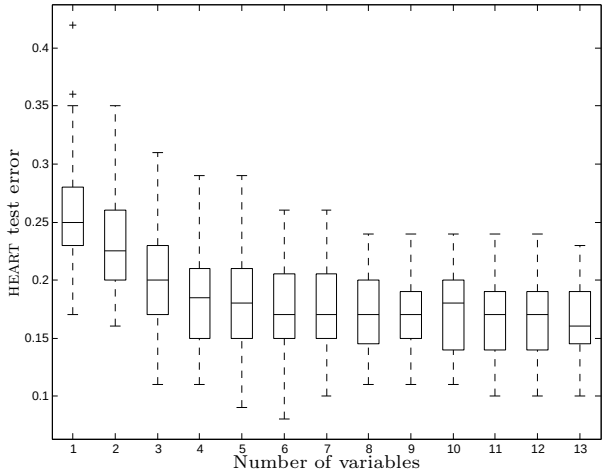


Figure 3.9: Boxplot for the HEART data set. Test error is averaged over 100 realizations. We use the LS-SVM with RBF kernel in a forward selection search. Hyperparameters (σ, γ) are tuned with the PRESS statistic.

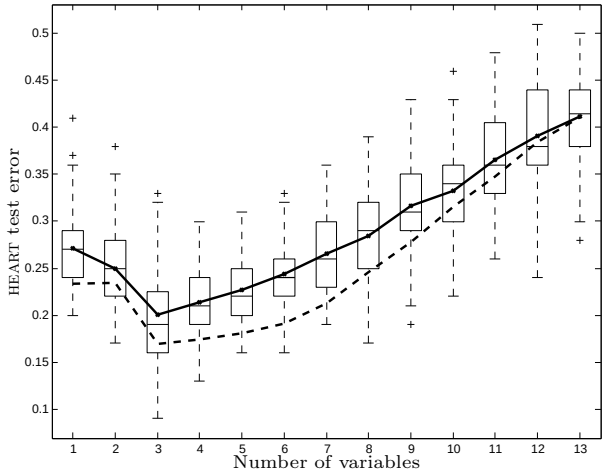


Figure 3.10: Boxplot for the HEART data set. Test error (solid line) is averaged over 100 realizations. We use the LS-SVM with RBF kernel. Hyperparameters (σ, γ) are tuned with the modified PRESS statistic. We additionally show the Error estimation (*Dashed*).

3.7 Summary

In this chapter, the application of *low rank updates/downdates* to the structure (linear kernel) of the LS-SVM classifiers was found to be very suitable for variable selection wrapper-like algorithms. By exploiting the inherent block structure of the LS-SVM solution, an efficient method for computing the LOO error enabled us to evaluate the performance of potentially relevant subsets of variables. Furthermore, in the special case of the *linear kernel*, algorithms for fast *forward* and *backward search* are devised such that *matrix inversions* are at most completely avoided. Our proposed methodology proceeds then by *updating* the LS-SVM classifiers every time a variable is tested for inclusion (or removal) into the set of candidate variables. Such strategy constitutes a major improvement in the efficiency of the algorithms presented here. In particular, matrix inversions are avoided at any cost in the low *rank update* algorithm, while the *rank downdate* version requires only a single matrix inversion. Experiments on microarray data provide us with evidence of the particular usefulness of our methods in high dimensional scenarios. Moreover, the use of a continuous function in the LOO estimator, the PRESS statistic, enables us to use optimization routines in the model selection, contrary to grid search approaches. Yet, alternatives to the LOO estimator based on continuous functions may provide more refined sets of variables, especially for data sets with reduced number of data points. Comparison with other related algorithms delivered satisfactory results in terms of efficiency and generalization performance.

Chapter 4

Polynomial componentwise LS-SVM: variable selection using low rank updates

In this chapter we present a LS-SVM approach to estimate additive models as a sum of non-linear components. In particular, this chapter considers low rank *matrix modifications for componentwise polynomial kernels, which allow the factors of the modified kernel-matrix to be directly updated. The main concept refers to the use of a valid explicit* feature map *for polynomial kernels in an additive setting. By exploiting the structure of such feature map the model parameters of the classification/regression problem can be easily modified and updated when new variables are added. Therefore, the low rank updates constitute an algorithmic tool to efficiently obtain the model parameters once the system has been altered in some minimal sense. Such strategy allows, for instance, the development of algorithms for sequential variable ranking in high dimensional settings, while non-linearity is provided by the polynomial feature map. Moreover relevant variables can be robustly ranked using the closed form of the leave-one-out error estimator, obtained as a by-product of the low rank modifications.*

4.1 Introduction

Additive models are very useful techniques for approximating high dimensional nonlinear functions [44]. These methods are widely used as nonparametric techniques as they offer a compromise between flexibility, dimensionality and interpretation. Estimation of the nonlinear components is usually performed by the iterative backfitting algorithm. Basically, in each step part of the unknown components are fixed while optimizing the remaining components. Another category of techniques for non-linear classification and function approximation are those based on regularization and kernel methods. In this context, smoothing splines [113], Gaussian processes [60], Support Vector Machines [111] all represent non-parametric techniques attempting to overcome problem of the curse of dimensionality. We have seen in previous chapters, that the success of kernel based methods relies on the indirect construction of feature spaces through positive-definite kernel functions. The main concept in this chapter however, resides on the use of an *explicit* finite dimensional feature map $\varphi(\cdot) \in \mathbb{R}^{d_\varphi}$, for additive polynomial kernels. By exploiting the structure of this feature map, the model parameters of the LS-SVM method can be easily modified and updated when a new variable enters the current model. Such strategy allows, for instance, the construction of efficient algorithms for sequential variable ranking in high dimensional settings.

Our approach considers LS-SVM as core classifiers and in particular the componentwise LS-SVM proposed in [79, 80], where an additive structure is imposed as a sum of nonlinear components. The primal-dual derivations characterizing LS-SVMs for the estimation of the additive model result in a single set of linear equations with size growing in the number of data-points. Furthermore, the use of low rank modifications in the LS-SVM structure, previously discussed in Chapter 3, provides efficient algorithms for sequential variable ranking wherein model parameters are updated instead of recalculated [75]. Nevertheless, the analysis was restricted to the use of linear kernels. This chapter advances precisely in this direction. Firstly, we incorporate non-linearity to the componentwise LS-SVM using an explicit feature map for the polynomial kernel. Secondly, we demonstrate how the factors of the modified componentwise LS-SVM can be efficiently updated. Within the model specification three purposes are recognized: (i) sequential variable selection using componentwise LS-SVM, (ii) efficient model updating

for linear and componentwise polynomial kernels, (iii) model interpretation with the imposed additive structure.

This chapter is organized as follows. Section 4.2 presents the *componentwise* extension to LS-SVM models. Section 4.3 outlines the explicit feature map for polynomial kernels and shows how low rank matrices result from the imposed additive structure. To update the componentwise LS-SVM parameters, methods based on the Cholesky decomposition are given in section 4.4. Experimental results of the proposed methodology are presented in section 4.6.

4.2 Componentwise support vector machines

This section considers the class of models that are additive in every input. Let the superscript for x^k, denote the k-th component of an input vector $\mathbf{x} \in \mathbb{R}^d$, for all $k = 1, \ldots, d$. To represent the collection of a single component the following notation is used $\mathbf{x}^k = [x_1^k, \ldots, x_i^k, \ldots, x_n^k]^\top$. Following the LS-SVM formulation given in Chapter 2, the componentwise regression/classification model takes the form $f(\mathbf{x}) = \sum_{k=1}^d \mathbf{w}_k^\top \varphi_k(x^k) + b$, where for the k-th component $\varphi_k(\cdot) : \mathbb{R} \to \mathbb{R}^{d_\varphi}$ is a possibly infinite dimensional mapping and $\mathbf{w}_k \in \mathbb{R}^{d_\varphi}$ the corresponding vector of model parameters. Therefore, the penalized sum of squares function, in contrast to (2.19), becomes [79]

$$\min_{\mathbf{w}_k, b, e} \frac{1}{2} \sum_{k=1}^d \mathbf{w}_k^\top \mathbf{w}_k + \gamma \frac{1}{2} \sum_{i=1}^n e_i^2 \tag{4.1}$$

$$\text{s.t. } y_i = \sum_{k=1}^d \mathbf{w}_k^\top \varphi_k(x_i^k) + b + e_i \ , \ i = 1, \ldots, n \ .$$

Along the lines of standard LS-SVM, by taking the conditions for optimality and application of the kernel trick $K^k(x_i^k, x_j^k) = \varphi_k(x_i^k)^\top \varphi_k(x_j^k)$, the following set of linear equations is obtained

$$\left[\begin{array}{c|c} \boldsymbol{\Omega}^d + \gamma^{-1}\mathbf{I}_n & \mathbf{1} \\ \hline \mathbf{1}^\top & 0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline b \end{array}\right] = \left[\begin{array}{c} \mathbf{y} \\ \hline 0 \end{array}\right] \ , \tag{4.2}$$

where as before $\boldsymbol{\Omega}^d = \sum_{k=1}^d \boldsymbol{\Omega}^k \in \mathbb{R}^{n \times n}$, and $\boldsymbol{\Omega}_{ij}^k = K^k(x_i^k, x_j^k)$, is the kernel evaluation at the k-th component between the i-th and the j-th data points. Note

that the main difference between the dual models in (2.20) and (4.2), is in fact expressed solely in terms of the used kernels [80]. Componentwise models of this type have been also used in the context of survival models [106]. A new data point $\mathbf{x} \in \mathbb{R}^d$ can be evaluated as

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i \sum_{k=1}^{d} K^k(\mathbf{x}^k, \mathbf{x}_i^k) + b\right) \ . \tag{4.3}$$

Remark 4.1. *Without loss of generality, the elements of the Gram matrices associated to kernel functions can be expressed as* $\mathbf{\Omega}_{ij} = \varphi(\mathbf{x}_i)^\top \varphi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$*, with* $i, j = 1, \ldots, n$*. By defining* $\Phi = [\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_n)] \in \mathbb{R}^{d_\varphi \times n}$*, the Gram matrix* $\mathbf{\Omega} \in \mathbb{R}^{n \times n}$ *becomes*

$$\mathbf{\Omega} = \begin{bmatrix} \varphi(\mathbf{x}_1)^\top \\ \vdots \\ \varphi(\mathbf{x}_n)^\top \end{bmatrix} [\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_n)] = \Phi^\top \Phi \ . \tag{4.4}$$

4.3 Polynomial updates

In this section, a more detailed description of the type of kernels considered in this work is given. The key idea is to view the overall kernel matrix as the sum of individual contributions of kernels when applied individually on every input. These kernels are denoted as *componentwise* kernels and in the case of polynomial functions, their structure provides low rank kernel matrices where their rank, for a univariate input, is determined by the polynomial degree. Starting from the linear kernel (2.10), the associated Gram matrix $\mathbf{\Omega}$ in (4.4) can alternatively be written in the form of outer products, that is

$$\mathbf{\Omega} = \left[\mathbf{x}^1, \ldots, \mathbf{x}^d\right] \begin{bmatrix} \mathbf{x}^1 \\ \vdots \\ \mathbf{x}^d \end{bmatrix}^\top = \sum_{k=1}^{d} \mathbf{x}^k \mathbf{x}^{k\top} = \mathbf{X}\mathbf{X}^\top \ , \tag{4.5}$$

with $\mathbf{X} = \left[\mathbf{x}^1, \ldots, \mathbf{x}^d\right]$. However, for more general kernels such as the polynomial or RBF kernel, the Gram matrix cannot be simply computed as the sum of outer products. Therefore, the family of kernels that are additive in

every input is instead considered. That is

$$K^d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{d} K^k(x_i^k, x_j^k) \ . \tag{4.6}$$

In this context, the individual kernels are defined on a single univariate input and hence functions of two scalar variables. The elements of the corresponding Gram matrix are $\mathbf{\Omega}_{ij}^k = K(x_i^k, x_j^k)$, that is, the kernel evaluated in the k-th component between the i-th and the j-th data points. The overall Gram matrix in this additive setting then takes the form [79]

$$\mathbf{\Omega}^d = \sum_{k=1}^{d} \mathbf{\Omega}^k \ . \tag{4.7}$$

It is clear that for the linear kernel $\mathbf{\Omega}^d = \mathbf{\Omega}$, that is the outer product definition given above in (4.5) and consequently also equivalent to that in (4.4).

For the polynomial kernel of degree p with $\tau = 1$ (cf. (2.11)), $K_p : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, a valid explicit feature map $\varphi_p(\cdot)$ for a given z is [17, 91]:

$$\varphi_p(z) = \left[1, \sqrt{\binom{p}{1}} z, \ldots, \sqrt{\binom{p}{p-1}} z^{p-1}, z^p\right]^\top \ , \tag{4.8}$$

with $\varphi_p(\cdot) : \mathbb{R} \to \mathbb{R}^{p+1}$. The Gram matrix for the additive polynomial kernel is then

$$\mathbf{\Omega}_p^d = \sum_{k=1}^{d} \mathbf{\Omega}_p^k \qquad \text{with } \mathbf{\Omega}_p^k = \sum_{l=0}^{p} (\varphi_l \circ \mathbf{x}^k)(\varphi_l \circ \mathbf{x}^k)^\top \ , \tag{4.9}$$

where $(\varphi_l \circ \mathbf{x}^k)$ corresponds to the elementwise application of the function $\varphi_l = \sqrt{\binom{p}{l}} z^l$ to the elements of vector $\mathbf{x}^k$. In matrix notation it reads

$$\mathbf{\Omega}_p^k = \Phi_p^k {\Phi_p^k}^\top \ , \tag{4.10}$$

where $\Phi_p^k = \left[(\varphi_0 \circ \mathbf{x}^k), \ldots, (\varphi_p \circ \mathbf{x}^k)\right]$ is a $n \times (p+1)$ matrix. Since for a Gram matrix the following holds

$$\text{rank}\left(\Phi_p^k\right) = \text{rank}\left(\Phi_p^k {\Phi_p^k}^\top\right) = \text{rank}\left(\mathbf{\Omega}_p^k\right) \ , \tag{4.11}$$

then for the additive model based on the polynomial kernel, the rank of $\mathbf{\Omega}_p^k$ for an input variable $\mathbf{x}^k$ is at most $p+1$. As a consequence, the polynomial kernel matrix $\mathbf{\Omega}_p^d$ for all inputs, $k = 1, \ldots, d$, and constant degree p can be written as the sum of d rank-$(p+1)$ matrices

$$\text{rank}\left(\mathbf{\Omega}_p^d\right) = \text{rank}\left(\sum_{k=1}^{d} \mathbf{\Omega}_p^k\right) \leq \sum_{k=1}^{d} \text{rank}\left(\mathbf{\Omega}_p^k\right) \quad . \tag{4.12}$$

The use of these componentwise (additive) models for input variable ranking is presented in the following sections. Moreover efficient algorithms are proposed along the lines of the earlier work in [75].

Remark 4.2. *Rank-one linear kernel. Setting $\tau = 0$ and $p = 1$ in the inhomogeneous polynomial kernel (cf. (2.11)) gives the linear kernel and therefore the rank of the mapping $\Phi(\mathbf{x}^k)$ for a single variable $\mathbf{x}^k$ equals 1 [75].*

Remark 4.3. *Rank-one homogeneous polynomial kernel. For a given $p \in \mathbb{Z}^+$ and setting $\tau = 0$, the inhomogeneous polynomial kernel (cf. (2.11)) reduces to the homogeneous kernel [92]. The rank of the matrix $\Phi_p(\mathbf{x}^k)$ for a single variable $\mathbf{x}^k$ reduces also to 1, for the lower order monomials receive a zero weighting coefficient.*

4.4 Low rank modifications

The system of linear equations in (4.2) can be alternatively factorized into a positive-definite system and hence the Cholesky decomposition can be used to obtain its solution [39]. In order to obtain the model parameters $\boldsymbol{\alpha}, b$, the solution of the two triangular systems $\mathbf{L}\mathbf{L}^\top \boldsymbol{\chi} = \mathbf{y}$ and $\mathbf{L}\mathbf{L}^\top \boldsymbol{\nu} = \mathbf{1}$, is required. The matrix $\mathbf{L}$ denotes the Cholesky factor of the positive-definite matrix $\mathbf{L}\mathbf{L}^\top = \mathbf{\Omega} + \gamma^{-1}\mathbf{I}_n$. The model parameters in (4.2) are then given by

$$b = \mathbf{1}^\top \boldsymbol{\chi} (\mathbf{1}^\top \boldsymbol{\nu})^{-1} \quad , \qquad \boldsymbol{\alpha} = \boldsymbol{\chi} - b\boldsymbol{\nu} \quad . \tag{4.13}$$

Adding a new variable $\mathbf{x}^k$ to the system of linear equations represents, in the simplest case, a *rank-one* modification to the Cholesky factor $\mathbf{L}$ [38] (see also Section 3.4)

$$\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top = \mathbf{L}\mathbf{L}^\top + \mathbf{x}^k \mathbf{x}^{k\top} \quad . \tag{4.14}$$

In the same line if the polynomial mapping $\varphi_p(\cdot)$ of the variable $\mathbf{x}^k$ is considered, then according to (4.10) and (4.11), the *rank-(p + 1)* modification of the Cholesky factor reads

$$\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top = \mathbf{L}\mathbf{L}^\top + \Phi_p^k \Phi_p^{k\top} \ . \tag{4.15}$$

Low rank updates of this form can always be done by applying $(p+1)$ rank-one updates sequentially with the columns of Φ_p^k. For the special *rank-one* case denote $\mathbf{u}$ as the variable $\mathbf{x}^k$ in the linear kernel, or a single column of Φ_p^k in the polynomial kernel. The modified Cholesky factor is obtained as follows

$$\begin{aligned} \tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top &= \mathbf{L}\mathbf{L}^\top + \mathbf{u}\mathbf{u}^\top \\ &= \mathbf{L}(\mathbf{I} + \mathbf{q}\mathbf{q}^\top)\mathbf{L}^\top \\ &= \mathbf{L}\bar{\mathbf{L}}\bar{\mathbf{L}}^\top\mathbf{L}^\top \ , \end{aligned} \tag{4.16}$$

where $\mathbf{q}$ is the solution to the triangular system $\mathbf{L}\mathbf{q} = \mathbf{u}$, while the Cholesky factor of the elementary matrix $\mathbf{I} + \mathbf{q}\mathbf{q}^\top$ is denoted by $\bar{\mathbf{L}}$. Since the product of two lower-triangular matrices is also a lower-triangular matrix, then the matrix $\mathbf{L}\bar{\mathbf{L}}$ is consequently the triangular factor required. The modified matrix $\tilde{\mathbf{L}}$ can be directly computed from $\mathbf{L}$ [38], which is simpler and cheaper to compute than starting from scratch. Therefore, the updated model parameters can be obtained from:

$$\tilde{b} = \mathbf{1}^\top\tilde{\boldsymbol{\chi}}(\mathbf{1}^\top\tilde{\boldsymbol{\nu}})^{-1} \ , \qquad \tilde{\boldsymbol{\alpha}} = \tilde{\boldsymbol{\chi}} - \tilde{b}\tilde{\boldsymbol{\nu}} \ . \tag{4.17}$$

where $\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top\tilde{\boldsymbol{\chi}} = \mathbf{y}$ and $\tilde{\mathbf{L}}\tilde{\mathbf{L}}^\top\tilde{\boldsymbol{\nu}} = \mathbf{1}$. For a positive update $\tilde{\mathbf{L}}$ remains positive-definite ($\tilde{\mathbf{L}} \succ \mathbf{0}$) if the initial $\mathbf{L}$ is also positive definite. On the other hand, negative downdates may break down if $\mathbf{L}\mathbf{L}^\top - \mathbf{u}\mathbf{u}^\top$ is not positive definite, which may lead to numerical errors and instability [39]. In this chapter we restrict ourselves to positive updates. Similar versions of the algorithms presented here can be derived using stable downdating procedures [38] and along the lines of Chapter 3.

4.5 Efficient leave-one-out computation

We have seen in Section 3.3 that the residual for the leave-one-out prediction of the i-th training data point is [19, 109]:

$$r_i^{(-i)} = y_i - \hat{y}_i^{(-i)} = \frac{\alpha_i}{(\mathbf{A}^{-1})_{ii}} \ , \tag{4.18}$$

where the diagonal elements of $\mathbf{A}^{-1}$ are as defined in (3.9). However, computing the leave-one-out error from the Cholesky factor $\mathbf{L}$ of $\mathbf{H}$ comprises numerically a more stable approach to obtain the residuals $r_i^{(-i)}$. Let $\mathbf{S} = \mathbf{L}^{-1}$ be the (lower triangular) inverse of the Cholesky factor, thus $\mathbf{H}^{-1} = \mathbf{S}^\top \mathbf{S}$. The diagonal elements of $\mathbf{A}^{-1}$ are then expressed (see [19] for more details).

$$(\mathbf{A}^{-1})_{ii} = \sum_{j=1}^{i} (\mathbf{S}_{ij})^2 + \frac{\nu_i^2}{s}, \ i = 1, \ldots, n \ . \tag{4.19}$$

The implementation of the proposed approach is outlined in Algorithm 4. In step 9, the MATLAB routine `cholupdate` or the code fragment in [38, pg. 43] can be used to update the Cholesky factor. Alternatively, the methods exposed earlier in Section 3.4.1 that make use of the SWM formula are also valid to update the model parameters.

4.6 Experimental results

In this section, experimental results illustrate the use of low rank updates when using componentswise LS-SVM classifiers and polynomial kernels. This approach overcomes, and reduces the high computational cost incurred during variable selection. Synthetic data and the benchmark dataset collected by [87] are employed to compare the usefulness of the methodology hereby exposed.

4.6.1 Model selection

The final performance of any learning strategy might be heavily influenced by the hyperparameter selection, i.e. a correct choice for (γ, τ). These hyperparameters are tuned as to minimize the PRESS statistic. For each

realization of the data sets, the pair (γ^*, τ^*) minimizing the leave-one-out PRESS statistic is chosen. In turn, that pair is used during the variable selection procedure. The model selection is performed independently for each realization of the data set, such that the standard errors would reflect the variability of the training algorithm and the model selection with respect to changes in the sampling of the data.

Algorithm 4 Variable ranking with componentwise polynomial kernels and low rank updates.

Input: Training set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$, regularization and polynomial kernel hyperparameters (γ, p, τ), number of variables to rank m. **Output:** Ranked variables $\mathcal{S}$, ranking criteria J.

Initialize set of active variables $\mathcal{A} = \{\mathbf{x}^1, \ldots, \mathbf{x}^d\}$.
Initialize set of ranked variables $\mathcal{S} = \emptyset$.
Initialize Cholesky factor $\mathbf{L}_0 = \text{diag}(\sqrt{1/\gamma}, \ldots, \sqrt{1/\gamma}) \in \mathbb{R}^{n \times n}$.
for $l = 1$ to m **do**
  for $k = 1$ to $|\mathcal{A}|$ **do**
    $\mathbf{L}_k \leftarrow \mathbf{L}_0$
    Compute for $\mathbf{x}^k \in \mathbb{R}^{n \times 1}$, the feature map $\Phi_p \in \mathbb{R}^{n \times p+1}$ using (4.8).
    for $r = 1$ to $p + 1$ **do** {For every column of Φ_p}
      Update Cholesky factor $\mathbf{L}_k \leftarrow \texttt{cholupdate}(\mathbf{L}_k, \Phi_p(:, r))$.
    end for
    Solve for $\boldsymbol{\alpha}$ using (4.13).
    Compute $\text{error}_{\text{press}}(k)$ using (4.19) and (4.18).
  end for
  Select variable $\mathbf{x}^c$ such that $c = \text{argmin}_{k \in \{1,\ldots,|\mathcal{A}|\}} \text{error}_{\text{press}}(k)$.
  Ranking cost $J(l) = \text{error}_{\text{press}}(c)$.
  Ranked variables $\mathcal{S} \leftarrow \mathcal{S} \cup \mathbf{x}^c$.
  Active variables $\mathcal{A} \leftarrow \mathcal{A} \setminus \mathbf{x}^c$.
  Set Cholesky factor for next iteration $\mathbf{L}_0 \leftarrow \mathbf{L}_c$.
end for
return $\mathcal{S}, J$

4.6.2 Toy Data

A synthetic toy data set drawn from an uniform distribution is generated according to the model $y_i = 10\,\mathrm{sinc}(x_i^1) + 20(x_i^2 - 0.5)^2 + 10x_i^3 + 5x_i^4 + \varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(0,1)$ [44]. A data set with $n = 100$ data points and $d = 500$ variables is generated to test the algorithm computational efficiency. For several degrees of the polynomial kernel ($p = 1, \ldots, 10$) the total time needed to compute 50 updates is measured. In Figure 4.1, each update corresponds to a new variable $\mathbf{x}^j,\ j = 1, \ldots, d$, added to set of linear equations. In addition, comparisons with respect to the time to compute the inverse from scratch (dotted line) are shown. A first alternative for efficiency is the use of the eigendecomposition method (dashed line) for the componentwise additive polynomial kernel. The eigendecomposition of the kernel matrix for each variable is first computed, and then a direct *rank* $(p+1)$ update is performed to obtain the inverse. The second method (solid line) corresponds to the main methodology introduced in this work, that is, the explicit feature map of the polynomial kernel for a single variable. Using this product form factorization, one can also directly apply a rank $(p+1)$ update to compute the inverse. Figure 4.2, presents the average LOO PRESS with respect to the number of variables ranked and the degree of the polynomial kernel.

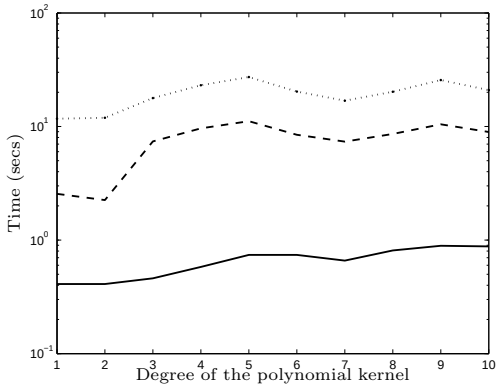


Figure 4.1: Computational time for the updates of the regularized Gram matrix when varying the degree of the componentwise additive polynomial kernel. Methods: full inverse (dotted line), eigendecomposition (dashed line), proposed (solid line).

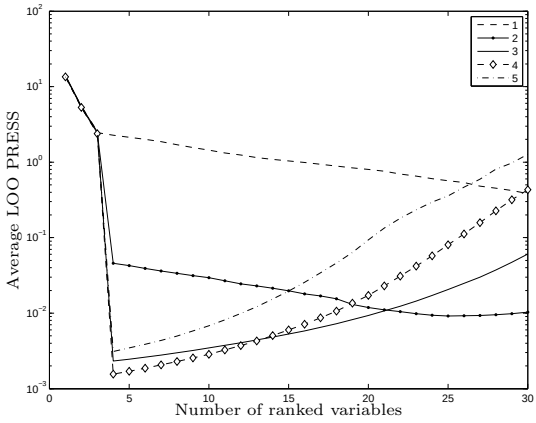


Figure 4.2: Leave-one-out PRESS for the synthetic data when different degrees in the componentwise polynomial kernel ($p = 1, \ldots, 5$) are tested. In all but two of the different models ($p = 1, 2$) the minimum averaged LOO PRESS value is attained at 4 variables, which in turn corresponds to the true number of components. Moreover, the polynomial kernel of degree 1 retrieves one spurious variable.

In a second experiment a number of 100 different training and test set partitions are simulated. The size of the data set n is varied from 10 to 100, and it is randomly chosen from the interval $[0, 1]^{100}$, thus $d = 100$. The degree of the polynomial kernel is varied $p = 1, 2, 3$ while $\tau \in \{0, 1\}$. In order to avoid some bias in the sequential forward algorithm, the 4 relevant variables are placed at random positions. In this setting, the goal is to check the capacity of the algorithm to retrieve the relevant variables giving them the highest rank. The regularization parameter γ is fixed to a value of 1 and the results are presented in Table 4.1. In the cases that τ is fixed to zero and the polynomial degree is higher, the lower order terms are discarded. The resulting non-linear mapping may not obey order restrictions, that is linear terms might be dropped while corresponding higher order terms are selected into the model. In fact the best entry for $\tau = 0$ corresponds to that of the linear model [75].

4.6.3 Real world data

In this section a selection of eleven UCI public domain benchmark data sets used in [87] are considered to evaluate the usefulness of the methodology here exposed. The use of low rank updates/downdates is demonstrated for linear and additive polynomial kernels. The ranking criterion for the variables is based on the PRESS statistic, which is derived from the efficient leave-one-out estimator [19, 21, 75]. Results obtained using standard LS-SVM and LS-SVM with automatic relevance determination (ARD) are taken from [21]. For standard SVM with different bounds, results were collected (though not complete) from previous studies in [22, 24, 85]. Results for each method are presented in the form of the mean error rate over test data for 100 realizations of each dataset (20 in the case of the image and splice datasets), along with the associated standard deviation of the error. Table 4.2 summarizes the results obtained, while Figure 4.3 illustrates the mechanism selecting the number of variables based on the LOO PRESS criterion.

	Algorithm	Training set size					
		10	20	30	40	50	100
$\tau = 0$	LowR-lin	44.64 (23.77) 17.5%	**9.85** (**12.05**) **61**%	3.96 (2.15) 74.75%	3.61 (2.94) 74.25%	3.11 (0.27) 75.75%	2.72 (0.12) 75.25%
	LowR-poly2	106.64 (147.30) 6.25%	60.66 (25.21) 7.25%	44.58 (12.23) 11%	37.96 (7.61) 13.25%	34.50 (5.75) 12.50%	26.28 (2.80) 23.25%
	LowR-poly3	113.52 (182.46) 13.75%	35.75 (19.06) 30%	16.87 (10.50) 52.50%	10.69 (3.10) 63%	9.19 (1.74) 69.25%	7.34 (0.78) 74.25%
$\tau = 1$	LowR-lin	**40.97** (**24.21**) **17.50**%	10.44 (12.96) 60%	4.44 (3.78) 71.50%	3.25 (0.36) 75.50%	3.07 (0.26) 75%	2.73 (0.17) 76.25%
	LowR-poly2	75.45 (89.61) 12.50%	21.04 (26.87) 57%	**1.14** (**5.76**) **96.25**%	0.19 (0.93) 99%	**0.05** (**0.01**) **100**%	0.05 (0.00) 100%
	LowR-poly3	120.00 (221.74) 8%	40.70(39.84) 39.75%	7.11 (15.05) 82%	**0.06** (**0.57**) **99.75**%	0.05 (0.47) 99.75%	**0.00** (**0.00**) **100**%

Table 4.1: Synthetic problem. Average generalization performance over 100 partitions of the data set. Standard deviations are indicated between brackets. The capacity to recover the four relevant components in this non-linear problem is expressed as the average success rate over 100 partitions. The larger the percentage value, the more variables were correctly recovered.

	LS-SVM		LowR LS-SVM			SVM		
Kernel	RBF		Linear	Polynomial		RBF		
Algorithm	LS-SVM	LS-SVM ARD	LowR-lin	LowR-poly2	LowR-poly3	$R\|\mathbf{w}\|_2^2$	Span	$\|\mathbf{w}\|_2^2$
BREAST	**26.73** (**0.47**)	29.08 (0.41)	27.88 (5.14)	33.34 (4.82)	34.06 (4.90)	26.84 (4.71)	**25.59** (**4.18**)	–
DIABETES	23.34 (0.17)	24.35 (0.19)	23.55 (1.71)	25.46 (2.20)	25.45 (1.93)	**23.25** (**1.70**)	**23.19** (**1.67**)	28.50
FLARE	34.22 (0.17)	34.39 (0.19)	**33.49** (**1.70**)	33.82 (1.65)	**33.31** (**1.78**)	–	–	–
GERMAN	**23.55** (**0.22**)	26.10 (0.26)	**24.97** (**2.17**)	28.29 (2.69)	28.89 (2.59)	–	–	–
HEART	16.64 (0.36)	23.65 (0.35)	**16.41** (**3.14**)	16.83 (3.25)	17.85 (3.72)	**15.92** (**3.18**)	16.13 3.11	27.00
IMAGE	**3.00** (**0.16**)	**1.96** (**0.11**)	18.44 (0.72)	10.56 (1.03)	7.99 (0.78)	–	–	–
RINGNORM	**1.61** (**0.01**)	**2.11** (**0.04**)	25.43 (0.55)	5.48 (0.54)	6.18 (0.53)	–	–	8.40
SPLICE	10.97 (0.16)	**5.86** (**0.18**)	16.13 (0.66)	10.55 (0.41)	**6.91** (**0.50**)	–	–	–
THYROID	4.68 (0.23)	4.68 (0.20)	20.24 (5.05)	18.57 (4.61)	11.87 (4.13)	**4.62** (**2.03**)	**4.56** (**1.97**)	–
TWONORM	2.84 (0.02)	5.18 (0.07)	**2.53** (**0.18**)	**2.67** (**0.21**)	3.43 (0.27)	–	–	9.30
WAVEFORM	**9.79** (**0.04**)	13.56 (0.14)	14.69 (0.10)	**11.13** (**0.57**)	11.41 (0.56)	–	–	–

Table 4.2: Benchmark data: Error rates of the compared methods. The results of each method are presented in the form of the mean error rate averaged over 100 test data realizations. Standard deviations are indicated between brackets. $-$: results not available in the literature.

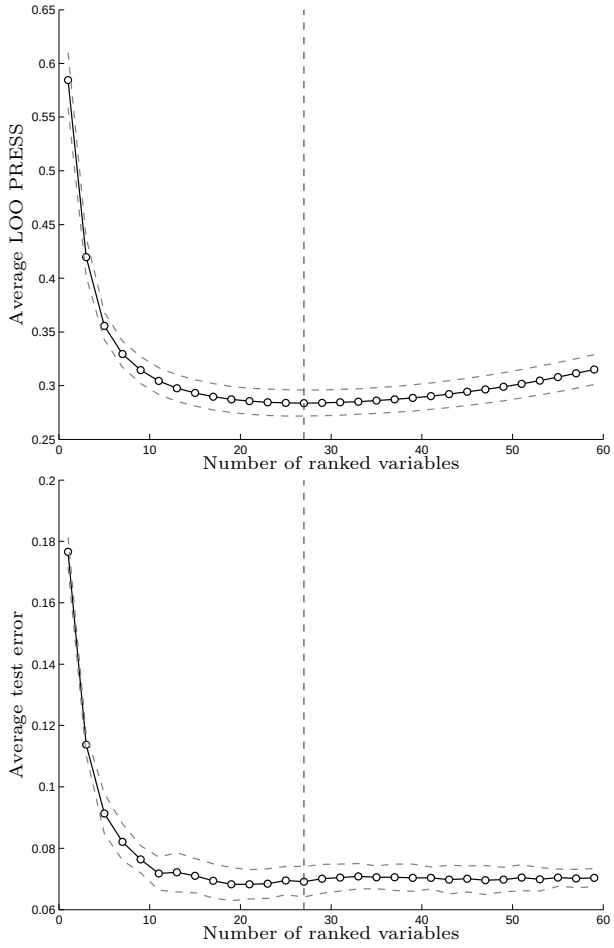


Figure 4.3: Selecting the number of ranked variables in the Splice data set based on the averaged LOO PRESS. A number of 27 variables out of 60 attain the lowest LOO PRESS averaged over 100 partitions. The degree of the polynomial kernel is set to a value of 3, whit the hyperparameters (τ, γ) tuned as described in the section over model selection. Top: Average LOO PRESS over 100 realizations. Bottom: Average test error over 100 realizations. Dashed lines represent the standard error deviation. The vertical dashed line points the model that achieves the lowest LOO PRESS and its corresponding test error performance.

4.7 Summary

The work presented in [40] proposes to solve the SVM algorithm only once with the full set of variables. Although the ranking criterion used there depends on the weight norm in the cost function, the support vectors $\boldsymbol{\alpha}$ remain fixed throughout the whole variable ranking. Moreover, even thought such strategy reduces efficiently the computational complexity of the approach from n-QP problems to merely one, it does violate however the optimality conditions in the optimization problem. In this work, the core learning algorithm is the (componentwise) LS-SVM classifier that can be efficiently trained by solving just a single system of linear equations. The main concept presented here, however concerns the the use of a valid explicit *feature map* for polynomial kernels in an additive construction. By exploiting the structure of the feature map, it is demonstrated how the model parameters of the classification/regression problem can be easily modified and updated when new variables enter the current model, while still providing non-linear modeling capabilities. This is achieved by the use of low rank updates which in turn constitutes an algorithmic tool for the design of sequential variable ranking algorithms in high dimensional settings. Moreover relevant variables can be robustly ranked using the closed form of the leave-one-out (LOO) error estimator directly obtained as a by-product of the low rank modifications.

Chapter 5

Learning of Sparse Linear Models in Mass Spectral Imaging

We present an approach to learn predictive models and perform variable selection by incorporating structural information from mass spectral imaging data. We explore the use of a smooth quadratic penalty to model the natural ordering of the physical variables, that is the mass-to-charge ratios. Thereby, estimated model parameters for nearby variables are enforced to vary smoothly. Similarly to overcome the lack of labeled data, we model the spatial proximity among spectra by means of a connectivity graph over the set of predicted labels. We explore the usefulness of this approach in a mouse brain MSI data set.

5.1 Introduction

Mass spectral imaging is a developing technology that accomplishes the detection of biomolecules such as proteins, peptides, and metabolites directly from organic tissue while retaining their spatial origin [99]. Thus, MSI allows to study the spatial distribution throughout the tissue of any detectable molecule that falls within a specified molecular mass range [63]. A typical MSI experiment consists of a grid of measurement locations or pixels covering the tissue section, with an individual mass spectrum attached to each pixel. The resulting data structure can be considered as a three-dimensional array or tensor with two spatial dimensions (h and w) and a mass-over-charge (m/z) dimension as shown in Figure 5.1.

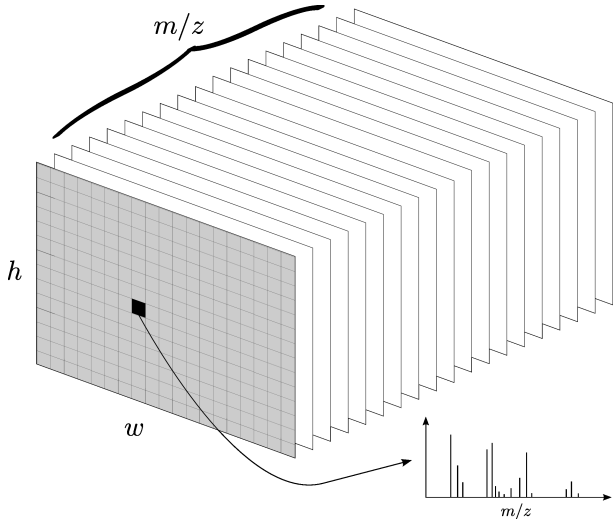


Figure 5.1: A schematic representation of the MSI data structure. Individual mass spectra are collected from the tissue area of interest retaining their spatial relationships (h,w). The data is collected into a three-mode array where each *slide* corresponds to a particular (m/z) value and every point in the grid is linked to a spectrum.

These characteristics pose challenges in the statistical analysis of MSI data. On the one hand, the high molecular specificity of MSI delivers huge dimensional data sets with thousands of measured variables that usually exceed the number

of spectra (data points), which are often limited to a few hundreds. On the other hand, the spatial coordinates (h, w) associated with each spectrum define physiological or anatomical areas of interest and their spatial relationships should thus not be neglected either. In practice several machine learning methods have already been applied to MSI data sets, including but not limited to principal component analysis [107], clustering and multivariate analysis [62], and supervised classification [42], [59]. Besides the relatively low number of spectra, only a small fraction of them is labeled. This hinders many statistical methods and further limits the validation of the obtained results. Manual labeling requires dedicated histological expertise which can be time consuming, costly and in some cases inaccurate.

In the present chapter, the goal is to develop semi-supervised models that use the labeled portions of the tissue to help predict the anatomical labels and biological functions of the unlabeled portions of the tissue. The medical objective of such models would be, for instance, to provide the pathologist with insight in interpreting molecular tissue content of areas that do not lend themselves for straightforward human classification. Particularly, we address issues regarding inherent ordering of the model variables and the spatial relationships of the samples. In a first step, we start from regularized models that impose sparsity on the solution of coefficients. In the problem of interest, variables possess a natural ordering due to their physical meaning. Therefore, we enforce that the estimated coefficients of nearby variables should smoothly vary in terms of m/z.

Unlike the so called *fused LASSO* [105] where the absolute value of the differences is used, we employ a smooth quadratic penalty. Furthermore, to overcome the lack of labeled spectra we exploit the prior assumption that nearby spectra are likely to have the same label. This is a meaningful assumption for many types of data: for instance a tumor is more likely to affect nearby cells than erratically affect disconnected regions of tissue. Our approach encodes the spatial proximity among spectra by means of a graph and hence can be seen as a semi-supervised method. The resulting proposed model is shown to be equivalent to a LASSO formulation and therefore can be efficiently solved via the LARS (Least Angle Regression) [28] algorithm. Each component in our optimization problem clearly embodies the structural information of MSI data, whereas regularization parameters trade off the complexity of the model in terms of sparsity, smoothness and unlabeled data points.

This chapter is organized as follows. Section 5.2 introduces the notions about regularized linear models and the required notation with respect to MSI data. The general concept of encoding structural information via the graph Laplacian is presented in Section 5.3, while Section 5.3.1 deals in detail with the modeling of the ordering of the m/z variables and the resulting optimization problem. Section 5.3.2, elaborates on the encoding of spatial information using unlabeled spectra and states the final proposed approach. Preliminary results on a mouse brain MSI data set are given in Section 5.4. Comparisons to related algorithms are reported along with visualization and interpretation of the obtained results.

5.2 Penalized regression

The MSI data set can be represented by a collection of n data points (spectra) measured over d variables (mass-to-charge ratios). The set of labeled spectra is $\mathcal{D}_\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, with $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, where y_i is the associated label to spectrum $\mathbf{x}_i \in \mathbb{R}^d$. Denote by x_i^k the k-th component of $\mathbf{x}_i$. Therefore $\mathbf{x}^k = [x_1^k, x_2^k \ldots, x_i^k, \ldots, x_n^k]^\top$ indicates the vector of measurements of a single variable collected at different locations across the tissue. We deal with the problem of predicting the response y, from a corresponding data point $\mathbf{x}$. In this setting, we consider the standard linear regression model

$$y_i = \sum_{k=1}^{d} w_k x_i^k + \varepsilon_i \ , \tag{5.1}$$

with errors $\varepsilon_i \sim \mathcal{N}(0, \hat{\sigma}^2)$. The variables are assumed to be standardized and the output to be centered. The vector of coefficients $\mathbf{w} = (w_1, \ldots, w_d)^\top \in \mathbb{R}^d$ is usually obtained by penalized empirical risk minimization:

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda P(\mathbf{w}) \ . \tag{5.2}$$

Common examples of penalized models are ridge regression with $P(\mathbf{w}) = \|\mathbf{w}\|_2^2 = \mathbf{w}^\top\mathbf{w}$, or the LASSO with $P(\mathbf{w}) = \|\mathbf{w}\|_1$. The LASSO penalty encourages sparse solutions while ridge regression keeps all the coefficients in the model. In general, a priori assumptions encoded via $P(\cdot)$ are needed to make the problem well-posed. In the following sections we aim at modeling the specific features of MSI data by translating them into useful structural information in the general optimization problem described in (5.2).

5.3 Structure Encoding via the Graph Laplacian

In order to incorporate structural information in our model fitting approach, we consider an undirected connectivity graph $\mathcal{G} = (V, E)$, where V is the set of nodes and E the set of edges. An edge between given nodes u and v ($u \neq v$) exists if the entities represented by u and v are linked. Denoting d_u as the degree of a node u, the normalized Laplacian matrix $\boldsymbol{\mathcal{L}}$ associated to the graph $\mathcal{G}$ is given by [23]

$$\boldsymbol{\mathcal{L}}(u,v) = \begin{cases} 1 & \text{if} \quad u = v \quad \text{and} \quad d_u \neq 0 \ , \\ - \ 1/\sqrt{d_u d_v} & \text{if} \quad u \neq v \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{5.3}$$

The Laplacian is a symmetric semi-positive definite matrix which can be interpreted as an operator on functions of the type $\mathbf{f} : V \to \mathbb{R}$ namely vectors indexed by elements of V. It can be shown that [23]

$$\mathbf{f}^\top \boldsymbol{\mathcal{L}} \mathbf{f} = \sum_{u \sim v} \left(\frac{f_u}{\sqrt{d_u}} - \frac{f_v}{\sqrt{d_v}} \right)^2 \ , \tag{5.4}$$

and hence the quadratic term on the left-hand side of (5.4) can be used to define a penalty enforcing smooth variation over neighboring nodes. We use this fact to incorporate structural information of MSI into the learning framework.

5.3.1 Encoding ordered variables

In order to account for the natural ordering of the m/z measurements, we impose a graph $\mathcal{G}^d$ over the set of variables. The set of nodes are associated to the d input variables $\mathbf{x}^k, k = 1, ..., d$, thus modeling neighboring variables via the Laplacian matrix of the graph. The structure imposed is visualized in the left panel in Figure 5.2, where every m/z variable $\mathbf{x}^k$ is connected to the preceding $\mathbf{x}^{k-1}$ and the subsequent $\mathbf{x}^{k+1}$. One might also consider second order relationships and so forth. By defining $\boldsymbol{\mathcal{L}}_\mathbf{w} \in \mathbb{R}^{d \times d}$ as the Laplacian over the set of variables (cf. (5.3)) and considering the squared norm in (5.4) for $\mathbf{w}$,

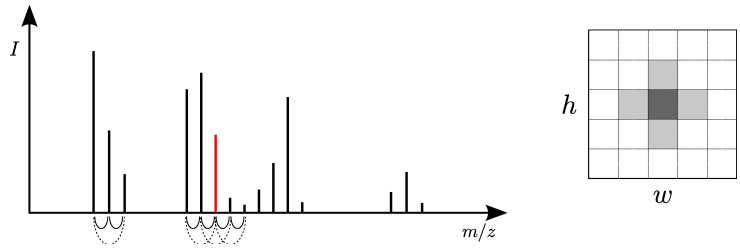


Figure 5.2: *Left*: First order (solid) and second order (dashed) connectivity structure over the set of the variables. We consider first order connectivity to impose local smoothness on the coefficients. *Right*: *Cross-like* spatial neighborhood imposed over the set of spectra.

our regularized optimization problem takes then the form:

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda_1 \sum_{k=1}^{d} |w_k| + \lambda_2 \mathbf{w}^\top \boldsymbol{\mathcal{L}}_{\mathbf{w}} \mathbf{w} \ , \tag{5.5}$$

with $\lambda_1, \lambda_2 > 0$. While the second term enforces sparsity on the $\mathbf{w}$, the last term smooths the solution of $\mathbf{w}$ on the network. This is similar to the formulations in [58] and [97]. In the case that no structure is assumed in the network, that is taking $\boldsymbol{\mathcal{L}}_{\mathbf{w}} = \mathbf{I}$, the optimization problem resorts to the elastic net (ENET) approach [123].

5.3.2 Encoding prior spatial information

Along the same lines of reasoning, we aim to impose a smooth structure on the predicted labels $\hat{y}_j,\ j = 1, ..., n_s$. The spatial distribution of the spectra in the square grid (see Figure 5.1) suggests that nearby spectra should correspond either to the same tissue area or, might represent connectivity tissues. In essence our goal is to extend the framework presented in the previous section by additionally incorporating the information of the spatial structure of the MSI data. In order to get an empirical estimate of spectra distribution we make use of the full set of examples: labeled and unlabeled spectra [14]. Denoting by $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_{n_s})^\top, n_s \geq n$, the vector of predicted responses and, assigning each $\hat{y}_j$ to a node in a graph $\mathcal{G}^s$, we construct the corresponding Laplacian matrix $\boldsymbol{\mathcal{L}}_s \in \mathbb{R}^{n_s \times n_s}$ using (5.3). The entries of $\boldsymbol{\mathcal{L}}_s(h, w)$ are defined according

to the *cross-like* neighborhood pattern shown on the right hand side of Figure 5.2. Likewise, we consider a similar quadratic form for the predicted responses as in (5.4), that is $\hat{\mathbf{y}}^\top \boldsymbol{\mathcal{L}}_s \hat{\mathbf{y}}$, which is bounded by parameter $\xi > 0$. Including this constraint into our optimization problem, we have

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda_1 \sum_{k=1}^{d} |w_k| + \lambda_2 \mathbf{w}^\top \boldsymbol{\mathcal{L}}_{\mathbf{w}} \mathbf{w} \tag{5.6}$$

$$\text{s.t. } \hat{\mathbf{y}}^\top \boldsymbol{\mathcal{L}}_s \hat{\mathbf{y}} \leq \xi \tag{5.7}$$

$$\hat{y}_j = \sum_{k=1}^{d} w_k x_j^k,\ j = 1, \ldots, n_s \ . \tag{5.8}$$

Expressing the vector of equality constraints in matrix form $\hat{\mathbf{y}}_s = \mathbf{X}_s \mathbf{w}$, and replacing this term in the inequality constraint we get $\mathbf{w}^\top (\mathbf{X}_s)^\top \boldsymbol{\mathcal{L}}_s (\mathbf{X}_s) \mathbf{w} = \mathbf{w}^\top \mathbf{G}_s \mathbf{w}$. By introducing a Lagrange multiplier $\lambda_3 > 0$ for the latter constraint, we write (5.6) as the following unconstrained optimization problem

$$\min_{\mathbf{w}} \ \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda_1 \sum_{k=1}^{d} |w_k| + \lambda_2 \mathbf{w}^\top \boldsymbol{\mathcal{L}}_{\mathbf{w}} \mathbf{w} + \lambda_3 \mathbf{w}^\top \mathbf{G}_s \mathbf{w} \ . \tag{5.9}$$

By grouping the quadratic terms of $\mathbf{w}$ and defining $\mathbf{H}_{\lambda_3} = \boldsymbol{\mathcal{L}}_{\mathbf{w}} + \frac{\lambda_3}{\lambda_2} \mathbf{G}_s$, we can further cast the optimization problem into a LASSO type formulation

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + | + \|\mathbf{0}_{d\times 1} - \sqrt{\lambda_2} \mathbf{H}_{\lambda_3}^{1/2} \mathbf{w}\|_2^2 + \lambda_1 \sum_{k=1}^{d} |w_k| \ \text{, or}$$

$$\min_{\mathbf{w}} \left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_{d\times 1} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda_2} \mathbf{H}_{\lambda_3}^{1/2} \end{bmatrix} \mathbf{w} \right\|_2^2 + \lambda_1 \sum_{k=1}^{d} |w_k| \ . \tag{5.10}$$

This modified problem has dimensions $(n + d) \times d$, and can be solved using the LARS (Least Angle Regression) algorithm [28] up to fixing λ_2 and solving the regularization path for the constrained version using a bound $\|\mathbf{w}\|_1 \leq \eta$ instead of λ_1. This equation is our final proposed method to incorporate spatial information and to impose neighboring structure in the ordered variables.

By setting $\lambda_3 = 0$, we can relate our method to similar existing approaches. For instance, the ENET [123] is obtained by setting $\boldsymbol{\mathcal{L}}_w = \mathbf{I}$. The network

constrained-regularization (NET) [58] and the multiple NET [97] are not restricted to ordered variables and instead they impose prior groupings through graphs. The fused LASSO algorithm in [105] penalized the absolute difference between adjacent coefficients whereas the the *group LASSO* [119] assumes in advance groups of variables.

5.4 Experimental results

In this section we explore the usefulness and applicability of the proposed method to include the structural information of MSI data. While numerical validation of the obtained results is assessed via 10-fold cross-validation, visual interpretation and comparison appear more intuitive by translating the results into exploratory ion images. This visual aid largely compensates for the limited availability of ground truth information.

5.4.1 Data set

The data set, acquired at University Hospital Leuven, comes from a sagittal section of mouse brain [108]. The spatial grid covering the tissue has 51 $\times$ 34 measurement locations (i.e. 1734 pixels). Each measurement spans a mass range from 2800 to 25000 Dalton in 6490 mass-to-charge (m/z) bins. Therefore, the data structure contains 1734 mass spectra measuring 6490 m/z variables per spectrum. Partial labeling information of 279 spectra is provided by a pathologist corresponding to four anatomical regions within the tissue. The labeled regions are the cerebellar cortex (cc), Ammon's horn in the hippocampus (ca), the cauda-putamen (cp), and the lateral ventricle (vl) area. Figure 5.3(a) depicts the four partially labeled regions overlaid on a gray level microscopic image of the mouse brain section. The set of spectra is normalized with respect to the total ion current and is baseline corrected.

5.4.2 Numerical results

In order to set suitable values for the three regularization parameters, we first define a grid of values over the parameters λ_2 and λ_3. Secondly, for every pair of values we approximate the regularization path for the parameter η (associated

to λ_1) and pick the best combination via 10-fold cross-validation. In Table 5.1, we report the results of the proposed approach among pairwise classes. Chosen values for the regularization parameters are reported along with the average 10-fold cross-validation accuracy and the number of non-zero $\mathbf{w}$ coefficients. Similarly, the performance of the LASSO and elastic net algorithms are reported in Table 5.2.

Classes	η^*	λ_2^*	λ_3^*	Non-zero $\mathbf{w}$	10-fold mse		10-fold accuracy	
cc vs *ca*	0.165	100	0.001	64	1.0529	(0.5482)	1	(0)
cc vs *cp*	0.213	100	0.001	106	1.3783	(0.3739)	0.9288	(0.1076)
cc vs *vl*	0.249	1	0.001	56	1.7588	(0.9709)	0.8938	(0.1719)
ca vs *cp*	0.162	1	0.001	54	2.3778	(0.9005)	0.9758	(0.0319)
ca vs *vl*	0.1640	100	0.01	102	3.2342	(1.2151)	0.9446	(0.0447)
cp vs *vl*	0.0460	10	0.1	14	5.9332	(1.6775)	0.9288	(0.0580)

Table 5.1: Multi-class one-vs-one results of the proposed approach. Regularization parameters associated to the quadratic penalties (λ_2, λ_3) are chosen from the grid $[10^{-3},\ 10^{-2},\ 10^{-1},\ 1,\ 10,\ 100]^2$. The regularization path for parameter η (bound on the ℓ_1 norm) associated to λ_1 is optimized via 10-fold crossvalidation on the labeled data.

	LASSO				ENET				
Classes	Non-zero $\mathbf{w}$	η^*	10-fold accuracy		Non-zero $\mathbf{w}$	η^*	λ_2^*	10-fold accuracy	
cc vs *ca*	14	0.29	1.0000	(0.0)	17	0.211	0.001	1.0000	(0.0)
cc vs *cp*	10	0.16	0.9738	(0.0532)	15	0.136	0.01	0.9905	(0.0202)
cc vs *vl*	34	0.374	0.9250	(0.1208)	17	0.145	0.01	0.9333	(0.1097)
ca vs *cp*	31	0.212	0.9740	(0.0436)	18	0.1	0.001	0.9687	(0.0477)
ca vs *vl*	11	0.085	0.9143	(0.0732)	19	0.076	0.1	0.9330	(0.0549)
cp vs *vl*	5	0.031	0.9142	(0.0739)	6	0.033	0.001	0.9123	(0.0766)

Table 5.2: Multi-class one-vs-one results of the LASSO and ENET algorithms.

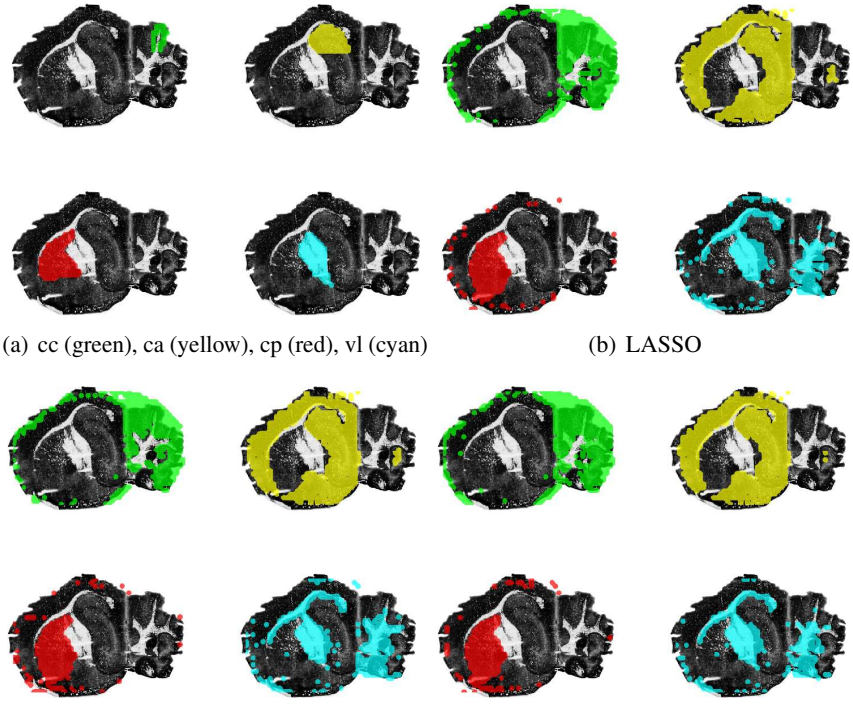
Additionally, the performance for the combined *one-versus-one* predictions is presented in Table 5.4.2. All the three methods perform slightly similar with appreciable differences in the average number of coefficients. The proposed method tends to select more coefficients due to the effect of the two square penalties.

Method	Avg. Non-zero $\mathbf{w}$	Avg. $\|\mathbf{w}\|_1$	10-fold accuracy
LASSO	17	1.0303	0.9453 (0.0690)
ENET	16	0.9608	0.9563 (0.0515)
proposed	66	1.6647	0.9502 (0.0608)

Table 5.3: Combined multi-class results on the mouse brain MSI data set.

5.4.3 Visualization

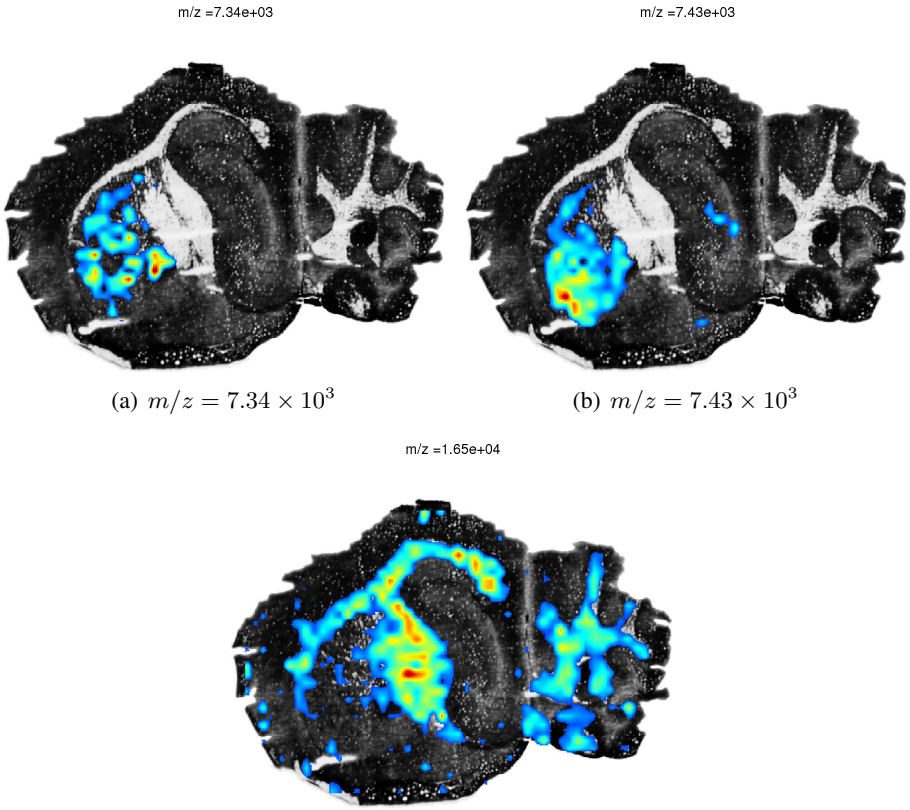
By translating the predicted labels back to their position in the spatial domain, one can directly assess the performance of the algorithm via visual inspection. Figure 5.3 displays the combined *one-vs-one* predicted labels corresponding to the compared methods. All the three models effectively separate the lateral ventricle (vl) and cauda-putamen (cp) from the surrounding tissue. The classification for the ventricle area additionally draws in the elongated corpus callosum and cerebellar nucleus regions as well, as these regions share a panel of common molecules within the measured mass range. The cerebellar cortex (cc) label exceeds its intended boundaries due to the small number of labeled spectra (21 data points). The remaining hippocampus label (ca) extends to capture the complete hippocampus and most of the remaining unlabeled areas of the tissue. Furthermore, to visualize important selected variables, we look at the top three variables associated to the largest $\mathbf{w}$ coefficients. In particular we take those differentiating the lateral ventricle (vl) from the cauda-putamen (cc). In Figure 5.4, ion images highlight the presence of three of the top selected m/z in these two anatomical regions.



(a) cc (green), ca (yellow), cp (red), vl (cyan) (b) LASSO

(c) ENET (d) proposed

Figure 5.3: Labeled areas and corresponding predicted labels by the algorithms. Visualization of the predicted tissue regions on the mouse brain MSI data set.



(a) $m/z = 7.34 \times 10^3$ (b) $m/z = 7.43 \times 10^3$

(c) $m/z = 1.65 \times 10^4$

Figure 5.4: Ion image visualization for the top three selected m/z variables discriminating the (cc) and (vl) tissue regions. The first two are common to the three compared methods, whereas the third variable at $m/z = 1.65 \times 10^4$ Da that delineates the (vl) area only appears in the proposed model.

5.5 Summary

In this chapter we have presented a methodology to learn semi-supervised sparse linear models in MSI data. Starting from regularized learning models and structural information inherent to MSI data, we make use of the graph Laplacian to embed first, the natural ordering of the m/z variables and, secondly the spatial location of the spectra. Thereby, smooth quadratic penalties are imposed over neighboring *nodes* representing in the first case *variables* and in the second one *data points*. These penalties modify the standard learning algorithm resulting in an equivalent LASSO formulation that can be solved efficiently. Moreover the lack of labeled data, typical of MSI experiments, is circumvented through modeling the predicted responses via the graph Laplacian. The applicability of the proposed approach is explored in a mouse brain MSI data set to distinguish amongst four anatomical regions, and it is compared to other learning models that do not, or partially, incorporate the structural information of MSI data. The presented case study shows that sparse linear models can already provide significant informative insight to assess tissue type, structure, and content. Additionally, our approach also holds value for more fundamental exploratory studies of tissue as it can highlight similarity in content between different tissue areas. Further work in this direction seems promising and should find applicability as more MSI data sets become available.

Chapter 6

Entropy based selection for spectral clustering

Although many clustering algorithms have been proposed for the analysis of gene expression data [98], the validity of the generated partitions is often assessed using the full data set. Therefore future test genes from a subsequent stage, lack of a model assigning them to one of the initial clusters. In this context, we explore the use of a particular spectral clustering formulation that incorporates an extension for out-of-sample points [9]. A clear advantage of the proposed method is the possibility to train and validate the clustering model [8, 9]. We make use of this algorithm to cluster genes in groups based on their expression values. Firstly, the clustering model is built upon a selected subset of informative genes and, in a second stage the model is used to infer cluster memberships for the remaining genes. Informative genes are selected via entropy maximization, related to the underlying density distribution of the data set. This subsampling scheme greatly reduces the computational burden in large number of genes while the out-of-sample extension provides a clustering model for new data points. Application on both synthetic and real gene expression data supports the usefulness of this approach.

6.1 Kernel *k*-means and Spectral Clustering

In contrast to the classification or regression tasks earlier discussed, the goal of clustering consists in partitioning a given data set into groups or *clusters* such that the data points belonging to the same group or cluster, will be more similar to each other than points in different clusters. This process is achieved by discovering groups and identifying interesting distributions and patterns in the underlying data in an *unsupervised* manner, that is by forming separated groups of data points within the *complete dataset*. In this scenario, given data $\mathcal{D} = \{\mathbf{x}_i^n\}$ the goal is to learn a function $f : \mathcal{X} \to \{1, \ldots, k\}$, which assigns each data point $\mathbf{x}_i$ to a corresponding cluster. The data set $\mathcal{D}$ is then broken down into a number of $k \leq n$ clusters $\mathcal{C} = \{\pi_1, \ldots, \pi_k\}$, where data points falling into the same cluster have a high within-cluster similarity and at the same time low similarity to all other data points not in the same cluster.

The assumption behind traditional linear clustering techniques is that the data space consists of elliptical regions. Consequently, these methods can not detect clusters that are non-linearly separable in the input space. The problem of non-linear separability of classes, can be circumvented by mapping the observed data to a higher dimensional space in a nonlinear way so that each cluster for each class unfolds into a simple form. A first approach to tackle non-linearity is kernel k-means [26, 121], where points are first mapped to a higher-dimensional feature space using a non-linear function, and the data points are then partitioned in this new space. *Spectral clustering* [8, 9, 69, 95] is another alternative that has strong connections with graph theory [23]. The core algorithm uses the eigenvectors of an affinity matrix to group the data points. These two methodologies can nevertheless be formulated in terms of kernel functions. Introducing these techniques in microarray data analysis allows for dealing with both high-dimensional data and nonlinear relationships in the data, since the kernel trick enables computations for high-dimensional input spaces.

6.1.1 The kernel *k*-means algorithm

Kernel k-means clustering can be considered as a generalization of the k-means clustering algorithm. As usual, we assume our original data to be mapped to some feature space $\mathcal{F}$ via the function $\varphi(\cdot) : \mathcal{X} \to \mathcal{F}$. Hereby, we follow the formulation earlier exposed in [83]. Given a set of data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$,

where each data point is represented in a d dimensional space, a smooth, continuous non-linear mapping $\varphi(\cdot)$ from the input space to the feature space, the goal is then to partition the data set into k clusters $\mathcal{C} = \{\pi_c\}_{c=1}^k$, while minimizing the within-cluster sum of squares, that is:

$$\min_{\mathcal{C}} \sum_{c=1}^{k} \sum_{i=1}^{n} I_{ic} \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}_c^{\varphi}\|_2^2 \ , \tag{6.1}$$

where $I_{ic} = 1$ if $\mathbf{x}_i \in \pi_c$ and zero otherwise. For each cluster, $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ represent the centers or centroids. Contrary to k-means, here the centroids are implicitly defined in the feature space as

$$\boldsymbol{\mu}_c^{\varphi} = \frac{1}{|\pi_c|} \sum_{i=1}^{n} I_{ic} \varphi(\mathbf{x}_i) \ ,$$

where $|\pi_c|$ denotes the size of clusters π_c. Application of the kernel trick results in

$$\begin{aligned} \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}_c^{\varphi}\|_2^2 =& K(\mathbf{x}_i, \mathbf{x}_i) - \frac{2}{|\pi_j|} \sum_{j=1}^{n} I_{jc} K(\mathbf{x}_i, \mathbf{x}_j) + \ldots \\ &+ \frac{1}{|\pi_c|^2} \sum_{j,l=1}^{n} I_{jc} I_{lc} K(\mathbf{x}_j, \mathbf{x}_l) \ . \end{aligned} \tag{6.2}$$

The key issue in extending traditional k-means to kernel k-means simplifies to computation of distance in the kernel induced space. Assigning a data point $\mathbf{x}_i$ to a cluster is

$$I_{ic} = \begin{cases} 1 & \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}_c^{\varphi}\|_2^2 \leq \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}_q^{\varphi}\|_2^2, \forall c \neq q, c, q = 1, \ldots, k \\ 0 & \text{otherwise} \ . \end{cases} \tag{6.3}$$

With respect to standard k-means clustering, the general structure of algorithm is preserved in the kernel version. Nevertheless, there are two main differences between both schemes, namely the non-linear mapping via the kernel trick and the lack of an explicit centroid in the feature space. The mapping from the feature space *back* to the input space is called the *pre-image* problem. Typically

the exact pre-image of a given $\varphi(\mathbf{x})$ does not exist and therefore one can only approximate it [55]. Instead, we consider a pseudo-centroid defined as the data point that is closest to the centroid in the feature space [83], i.e.

$$\min_{\boldsymbol{\mu}_c,\, \mathbf{x}_i \in \pi_c} \|\varphi(\mathbf{x}_i) - \boldsymbol{\mu}_c^{\varphi}\|_2^2 \,, (1 \leq c \leq k) \ . \tag{6.4}$$

Similar to standard k-means, convergence to the global optimum for kernel k-means is not guaranteed. The non-convexity nature of the optimization problem (6.1) is sensible to the initialization. Multiple runs are thus often required.

6.1.2 Spectral clustering

To overcome the non-convexity in greedy k-means algorithms, spectral clustering methods relax the original problem in a way that global optimal solutions can be found. Spectral clustering techniques can be regarded as relaxations of graph cut problems, with nodes representing data points and edges the pairwise similarity among them. Clustering then corresponds to partitioning the nodes in the graph into groups. Such a division of the graph nodes in two disjoint sets is called a *graph cut* [23].

In the special case of a partitioning of the data into two groups, we aim to learn a clustering function $f : X \to \{+1, -1\}$. The set of data points is represented by graph $\mathcal{G} = (V, E)$ and similarity matrix $\mathbf{W}_{ij} = w_{ij} \geq 0$, where w_{ij} can be interpreted as the weights on the edges connecting nodes i and j in a graph. For $w_{ij} = 0$, there is no edge from i to j, i.e. $\mathbf{W}$ plays the role of an adjacency matrix often taken as the RBF kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{W}_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\sigma^2)$.

In this way, the problem of clustering consists in minimizing the cost of *cutting* the graph into k disjoint sets $\pi_1, ..., \pi_k$. Let $d_i = \sum_j w_{ij}$, called the degree of node i, and $\mathbf{D} \in \mathbb{R}^{n \times n}$ a diagonal matrix with $\mathbf{D}_{ii} = d_i$. Furthermore, let $\mathbf{y} = [y_1, \ldots, y_n]^\top \in \{+1, -1\}^n$ a vector of labels that are assigned to the data items by the clustering algorithm. Then the cut-cost of splitting the graph in two partitions according to a given clustering is given as

$$\sum_{y_i \neq y_j}^{n} w_{ij} = \frac{1}{2}\left(\sum_{i,j=1}^{n} w_{ij} - \sum_{i,j=1}^{n} y_i y_j w_{ij}\right) = \frac{1}{2}\left(\mathbf{1}^\top \mathbf{W}\mathbf{1} - \mathbf{y}^\top \mathbf{W}\mathbf{y}\right) = \frac{1}{2}\mathbf{y}^\top (\mathbf{D}–\mathbf{W})\mathbf{y} \ .$$

(6.5)

In order to favor balanced clusters Shi and Malik [95] proposed the so-called normalized cut NCUT. Minimizing *cut*-like costs is known to be NP complete [23]. Nevertheless, allowing $\mathbf{y}$ to take real values, an optimal $\mathbf{y}^*$ can be computed as the solution of the following optimization problem:

$$\min_{\mathbf{y}} \mathbf{y}^\top(\mathbf{D}-\mathbf{W})\mathbf{y} \tag{6.6}$$

$$\text{s.t. } \mathbf{y}^\top\mathbf{D}\mathbf{y} = 1 \ .$$

The relaxed solution corresponds to the eigenvector associated to the second smallest eigenvalue of the generalized eigenvector problem

$$(\mathbf{D}-\mathbf{W})\mathbf{y} = \lambda\mathbf{D}\mathbf{y} \ . \tag{6.7}$$

Clustering is then performed by thresholding $\mathbf{y}^*$ appropriately. The matrix $\boldsymbol{\mathcal{L}} = \mathbf{D}-\mathbf{W}$, is the so-called Laplacian matrix of the graph. The top k eigenvectors provide relaxed clustering indicators $\mathbf{y} \in \mathbb{R}^n$. In this setting, however, there is no clear way as how test data points (out-of-sample) should be assigned to the initial clusters, since the embedding eigenvectors are only defined for the full data set. In [33] the authors employed the Nyström method in order to approximate eigenvectors and reduce the computational load for large scale applications. In contrast to this approach, the authors in [8,9] present the spectral clustering methods within a primal-dual optimization framework. This way, the clustering model obtained can be readily applied to new data. Such formulation arises from the context of Weighted Kernel PCA and LS-SVMs [100], where the projected variables $z_i = \mathbf{w}^\top\varphi(\mathbf{x}_i),\ i=1,\ldots,n$ have maximal variance and clusters structures become tighter. For a symmetric positive definite weighting matrix $\mathbf{V}$ (typically chosen to be diagonal) the following primal problem [8,9]

$$\min_{\mathbf{w},\mathbf{z}} \gamma\frac{1}{2}\mathbf{z}^\top\mathbf{V}\mathbf{z} - \frac{1}{2}\mathbf{w}^\top\mathbf{w} \tag{6.8}$$

$$\text{s.t. } \mathbf{z} = \Phi\mathbf{w} \ , \ \mathbf{V} = \mathbf{V}^\top > 0 \ , \tag{6.9}$$

where $\mathbf{z} = [z_1,\ldots,z_n]^\top$ is the compact form of the projected variables $z_i = \mathbf{w}^\top\varphi(\mathbf{x}_i)$, $\mathbf{V} = \text{diag}([v_1,\ldots,v_n])$ is the weighting matrix and $\Phi =$

$[\varphi(\mathbf{x}_1)^\top; ...; \varphi(\mathbf{x}_n)^\top]$ is the $n \times d_h$ feature matrix. An eigenvalue problem is obtained after elimination of primal variables

$$\mathbf{V}\boldsymbol{\Omega}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha} \ , \tag{6.10}$$

where $\boldsymbol{\Omega}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^\top\varphi(\mathbf{x}_j)$ and $\lambda = 1/\gamma$. Note that, if the weighting matrix $\mathbf{V}$ is set to the inverse of the graph degree matrix $\mathbf{V} = \mathbf{D}^{-1}$, the eigenvalue problem (6.10) is equivalent to the binary NCUT (c.f. (6.7)) and random walk spectral clustering [64]. For a new test point $\mathbf{x}$, its projection

$$z(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \ ,$$

defines the extension to out-of-sample data points. To project on more eigenvectors

$$z^{(r)}(\mathbf{x}) = \sum_{i=1} \alpha_i^{(r)} K(\mathbf{x}_i, \mathbf{x}), \ \ r = 1, \ldots, k \ . \tag{6.11}$$

The top k eigenvectors $\boldsymbol{\alpha}$, of $\mathbf{D}^{-1}\boldsymbol{\Omega}$ provide an embedding that contains the clustering information of the original data points. One of the main advantages of this formulation is the possibility to evaluate the model on out-of-sample data points. This is important for predictive capabilities, model selection and even large-scale data analysis. Other techniques such as the Nyström method approximate the eigenvectors for out-of-sample data points. In general, the relaxed solutions provided by spectral clustering methods are real-valued and do not provide cluster indicators. In [95], the authors propose to apply the normalized cut in a recursive way, that is the current partition is subsequently divided if the NCUT is below some pre-specified value. The process is repeated until k clusters have been found. Another alternative to obtain k clusters is called *re-clustering*. This approach consists of computing the top k eigenvectors of the eigenvalue problem in (6.7) or (6.10) and then performing k-means onto the eigenvector space. The general approach of spectral methods for clustering of genes expression data is summarized in Figure 6.1.

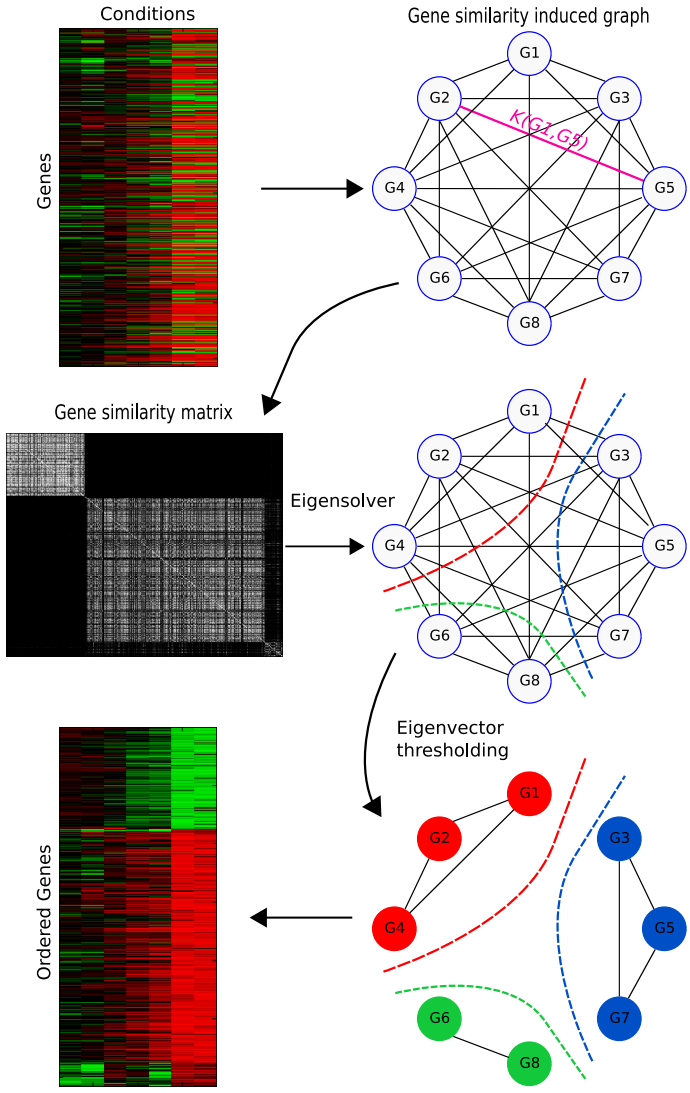


Figure 6.1: *Top row:* Gene similarities induce a graph. *Middle row:* Genes embedding via $K(\cdot, \cdot)$ similarity function. *Bottom row:* Eigenvectors or score variables provide clustering.

6.2 Selecting informative data points

A major drawback in kernel-based learning is that the amount of computation required to find the solution scales with the number of data points n. Hence, using the entire data set $\mathcal{D}$ to compute the clustering model becomes prohibitive for large values of n. Approximation to the eigendecomposition of the Gram matrix can be computed by the Nyström method (which is used for the numerical solution of eigenproblems) [117]. This is achieved by carrying out an eigendecomposition on a smaller system of size $m \ll n$, and then expanding the results back up to n dimensions.

In the context of estimation in the primal space [100, 116], it has been motivated to use a subset $m \ll n$ to compute an approximation of the feature map $\hat{\varphi}$. In general, the selection of the subset of size m, is performed before the model estimation. Here, instead of directly approximating $\hat{\varphi}$, we aim at selecting a subset of informative data points (genes) to construct the clustering model in (6.10). This approach leads to a reduced eigenvalue problem. In order to select the working subset, we seek to maximize the quadratic Renyi H_R [100]. In this case, given a fixed-size m, the goal is to select informative genes maximizing

$$H_S = -\log \int p(x)^2 \mathrm{d}x$$

$$H_S \approx H_R = \int \hat{p}(x)^2 \mathrm{d}x = \frac{1}{m^2} \mathbf{1}^\top \mathbf{\Omega}_h \mathbf{1} \ . \tag{6.12}$$

The use of this active selection procedure can be quite important for large scale problems, as it is related to the underlying density distribution of the data set. Here, $\mathbf{\Omega}_h$ is a $m \times m$ Parzen kernel estimator of width $\hat{h}$. The entropy criterion ensures that the selected subset is spread over the entire data region and not only concentrated on a certain area of the data set. The kernel bandwidth of the entropy based selection criterion is determined using a plug-in method as described in [25]. Criterion (6.12) can be iteratively maximized through the greedy strategy summarized in Algorithm 5.

Algorithm 5 Subset selection maximizing Renyi's entropy.

Input: Training set $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^d$, size of subset m. **Output:** Active set $\mathcal{A} = \{\mathbf{x}_j\}_{j=1}^m \subset \mathcal{D}$.

```
Select randomly an initial active set of size m, A = {x_j}_{j=1}^m ⊂ D.
repeat
    Compute H_R(A).
    Randomly pick two data points as x_* ∈ A, and x_+ ∈ D \ A
    Let W = {A \ {x_*}} ∪ {x_+}
    if H_R(W) > H_R(A) then
        swap(x_+, x_*)
    end if
until change in H_R value is too small
return A
```

It has been extensively reported that this strategy performs far better than random picking, often requiring a smaller working set [25, 30, 100]. Alternatively, similar to Radial Basis Networks (RBF) applications [68, 76], we also consider the k-means algorithm to select the working subset. That is, first we perform k-means clustering on the input space and secondly use the centroids (such as in equation (6.4)) as the working subset $\mathcal{A}$. The final proposed spectral clustering algorithm combined with working subset is described in Algorithm 6. For the re-clustering phase, we follow the k-means implementation in [29] which uses the triangle inequality to speed up computations[1].

6.2.1 Model selection

Opposite to standard clustering settings, we consider in all our simulations a full learning framework. That is, we made a clear distinction between training, validation and test phases. The number of clusters k and the kernel parameter σ^2 are chosen such that the resulting partition of data points into clusters $\mathcal{C} = \{\pi_1, \ldots, \pi_k\}$ is optimal with respect to a certain index. To validate our results we compute the Caliński and Harabasz (CH) index [18], which is reported to perform consistently among several other indexes [66]. The CH

[1] MATLAB code available at `http://cseweb.ucsd.edu/~elkan/fastkmeans.html`

Algorithm 6 Spectral clustering

Input: Training set $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^d$, active set $\mathcal{A} = \{\mathbf{x}_j\}_{j=1}^m \subset \mathcal{D}$. Number of clusters k. Kernel parameter σ^2. **Ouput**: Clustering partitions $\mathcal{C} = \{\pi_c\}_{c=1}^k$.

1: For $\mathcal{A}$, compute $\mathbf{\Omega}_{jl}{=}K(\mathbf{x}_j, \mathbf{x}_l) {\in} \mathbb{R}^{m\times m}, j, l = 1, \ldots, m$.
2: Obtain top k eigenvectors $\boldsymbol{\alpha}^{(r)} \in \mathbb{R}^{m\times 1}, r = 1, \ldots, k$ from (6.10).
3: $\forall \mathbf{x}_j \in \mathcal{A}$, compute the score variables $z_j^{(r)} = \sum_{l=1}^m \alpha_l^{(r)} K(\mathbf{x}_l, \mathbf{x}_i)$.
4: Denote $\mathbf{z}_j = [z_j^{(1)} \ldots z_j^{(k)}]$ and $\mathbf{Z} = [\mathbf{z}^{(1)} \ldots \mathbf{z}^{(k)}] \in \mathbb{R}^{m\times k}$, as the score variables space.
5: Perform standard k-means on $\mathbf{Z}$ to obtain cluster centroids $\boldsymbol{\mu}_c \in \mathbb{R}^k$, and partitions $\mathcal{C} = \{\pi_1, \ldots, \pi_k\}, c = 1, \ldots, k$.
6: Obtain $\hat{z}_i^{(r)} = \sum_{j=1}^m \alpha_j^{(r)} K(\mathbf{x}_j, \mathbf{x}_i)$, for out-of-sample data points, $\mathbf{x}_i \in \mathcal{D}$.
7: Assign $\mathbf{x}_i$ to $\pi_{c^*} \in \mathcal{C}$, where $c^* {=} \operatorname{argmin}_c \|\hat{\mathbf{z}}_i - \boldsymbol{\mu}_c\|_2^2$.
8: **return** $\mathcal{C} = \{\pi_c\}_{c=1}^k$.

index is defined as:

$$\mathrm{CH}(k) = \frac{\mathrm{trace}(\mathbf{S}_B)/(k-1)}{\mathrm{trace}(\mathbf{S}_W)/(n-k)} \ , \tag{6.13}$$

where $\mathbf{S}_B$ represents the between-cluster dispersion matrix and $\mathbf{S}_W$ the within-group dispersion matrix. The value of k, which maximizes $\mathrm{CH}(k)$, is regarded as specifying the number of clusters [31]. Contrary to the standard approach in clustering of selecting the parameters on the basis of the full data set, we make use of a randomly selected validation set. Using the out-of-sample extension in (6.11), we can readily compute the CH on the score variables space $\mathbf{Z}$. Additionally, we tune the value of the σ^2 parameter as well, thus choosing the pair (σ^{2^*}, k^*) that maximizes CH on the validation score variables.

A wide number of clustering quality measures [41] can be readily used in combination with the spectral clustering algorithm [83]. The only difference resides in that computation for such indexes is carried on the projected variables $\mathbf{Z}$ (also on eigenvectors [69, 83]), rather than in the original data set $\mathbf{X}$. In the case of kernel k-means, however, clustering indexes can not be directly used as they are defined for finite dimensional spaces. To evaluate clustering indexes in more general feature spaces, one must rewrite them first in terms of dot products.

In fact, we have previously extended the CH index along with the silhouette score [89] to more generalized versions in terms of kernel evaluations [83].

6.3 Experiments

In order to test the proposed methodology, we first create a set of toy examples both including linear and nonlinear clustering problems. Such problems allow us to observe the impact that the subsampling method, subset size m, number of clusters k and kernel parameter σ^2, have on the clustering output and to refine the general approach. Additionally, we test our methodology on a standard yeast microarray data set [16] to cluster genes based on their expression value. In general, we will observe that we can set $m < n$ without any significant decrease in the accuracy and quality of the clustering solution.

6.3.1 Toy data

We consider first a two-dimensional artificially generated data set with $k = 15$ pre-defined clusters and $n = 5000$ total number of data points. This data set $(\mathrm{S}1)^2$ presents a varying complexity in terms of data distribution, overlap and noise [34]. Secondly, we vary the subset size $m \in \{100, 300, 500, 1000\}$ and the number of clusters in the range $k = 2, \ldots, 20$. Finally, we calculate the average maximum CH index over 20 randomizations on validation data. In Figure 6.2, the original data set S1 is shown, along with the entropy selected data points. Clustering partitions $k = 2, 3, 15$, with maximum CH index are also illustrated. Optimal values of the CH index are consistently attained $k = 2, 3, 15$, hence indicating the multi-scale nature of the data set. After $k = 15$ clusters, there is an evident decrease in the index performance.

[2] http://cs.joensuu.fi/sipu/datasets/

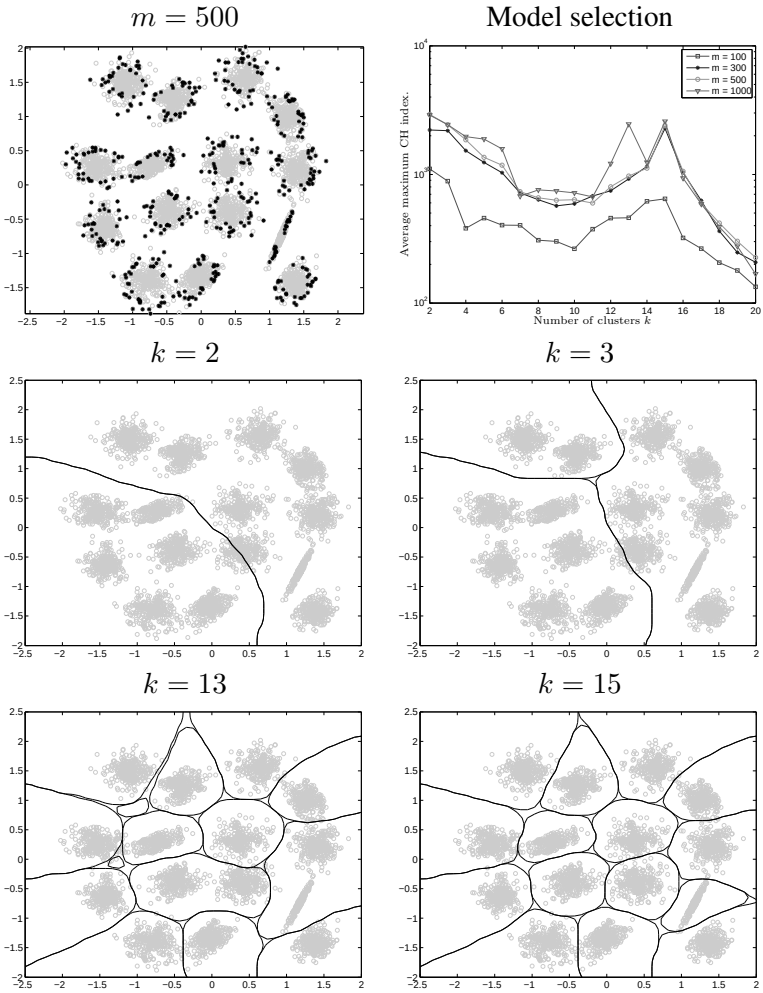


Figure 6.2: S1 data set clustering results. **Top row:**(*left*) Full data set and entropy selection with $m = 500$ data points. (*right*) Average maximal CH index for varying subset size m and number of clusters k. Optimal values are consistently attained at $k = 2, 3, 15$ clusters. Performance falls clearly off after $k = 15$ clusters. **Middle row:** Cluster partitions with average maximum CH index for $k = 2, 3$. **Bottom row:** Cluster partitions with average maximum CH index for $k = 13, 15$.

6.3.2 Non-linear cluster structures

In order to illustrate the benefit of using a kernel-based method and quantify whether the entropy subsampling is effective or not, we consider a classical non-linear data set in spectral clustering. The data set consists of a highly dense Gaussian cloud enclosed by two concentric rings-type clusters of lower density. We refer to this data set as CLOUD&RINGS. In the first stage, we vary the subset size from $m \in \{100, 300, 500, 1000\}$ from a total number of $n = 5000$ data points. Finally, we compare three subsampling methods namely: random, k-means and entropy, reporting performance in terms of the CH index and the adjusted rand index (ARI) with respect to the true clusters [49]. The expected value of the adjusted Rand index has value zero and the maximum value of the adjusted Rand index is 1, that is a perfect agreement between the produced clustering and the reference. Table 6.1 summarizes the obtained results and Figure 6.3 represents the data structure along with the selected data points.

m	random		k-means		entropy	
	CH	ARI	CH	ARI	CH	ARI
100	1.86E+05	0.531	2.16E+05	0.582	2.28E+05	1
300	2.17E+05	0.507	1.39E+05	0.588	3.04E+05	1
500	1.96E+05	0.566	2.1E+05	0.571	4.16E+05	1
1000	2.1E+05	0.561	1.99E+05	0.581	4.43E+05	1

Table 6.1: Numerical performance on the CLOUD&RINGS problem with with $n = 5000$, $k = 3$ and $\sigma^2 = 1.8$. The entropy subsampling attains a perfect agreement (ARI = 1) with as few as $m = 100$ data points. Neither random subsampling nor k-means reach good performance.

Clearly the entropy selection provides a superior estimation of the underlying data distribution, therefore allowing a better final clustering. The selected data points are nicely spread over the data set. Interestingly, the highly dense Gaussian cloud requires few number of data points to be represented, hence we could think of it as a compressed codebook. On the contrary, random and k-means subsampling are somewhat fooled by the Gaussian cloud as they select almost all the data points from this high density region. These two approaches fail thus at capturing the structure of the two rings (cf. Figure 6.3).

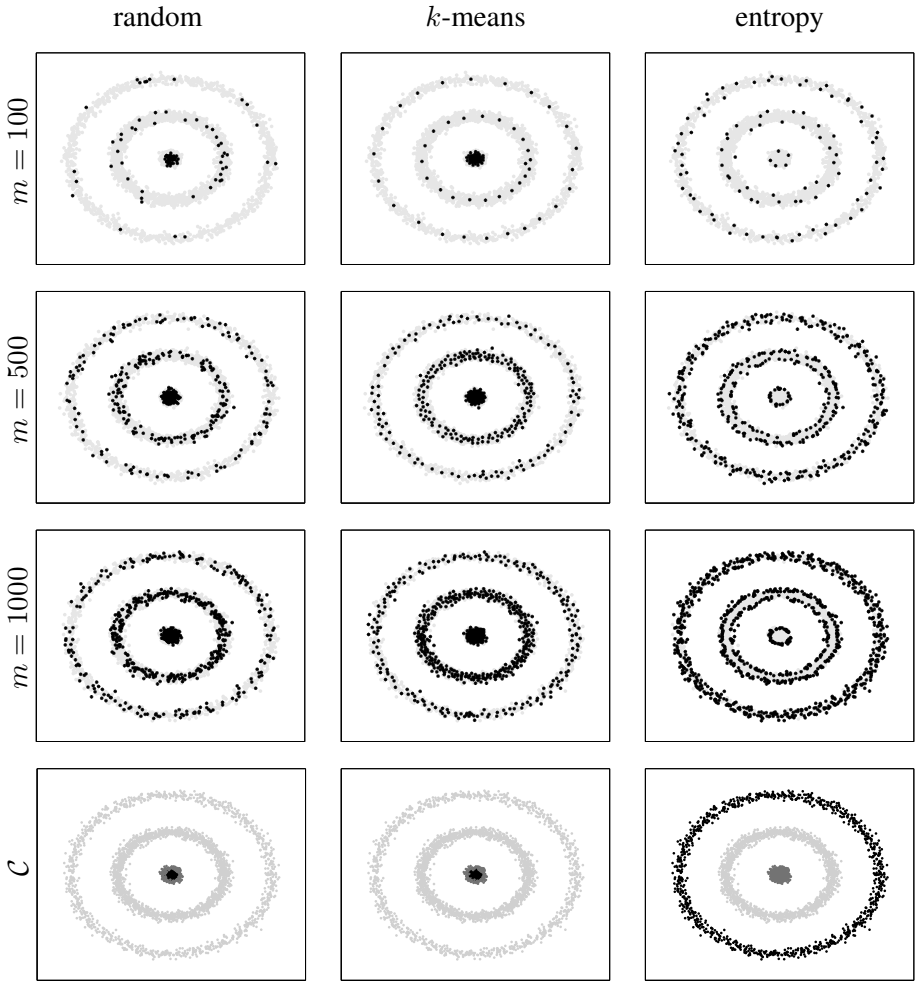


Figure 6.3: Comparison of three subsampling schemes for different subset size m on the CLOUD&RINGS problem. **Bottom row:** Best clustering result for each subsampling scheme.

6.3.3 Simulated and real data gene expression data

In order to test our kernel-based clustering approach, we consider a simulated data and a yeast galactose microarray data set that have been extensively reported in previous studies. The simulated data (SIMULATED1000) set consists of four clusters of 1000 genes under 100 conditions [65]. The yeast galactose microarray data set (YEASTG) represents a time series showing the expression profiles of yeast growing with 205 genes measured under 20 different perturbations for the GAL pathway [50]. Given that information over the true partitions is known, we employ the adjusted rand index (ARI) to compare our clustering approach against previously reported methods [120]. Table 6.2 summarizes the results of the synthetic data set for different subset sizes. We report the CH index for both the entropy subset and the final projected set of genes.

m	CH subset	CH projected	ARI
100	2.4E+10	3.3E+11	0.99
200	4.9E+10	3.3E+11	0.99
300	7.4E+10	3.3E+11	0.99
400	9.9E+10	3.3E+11	0.99
500	1.2E+11	3.3E+11	0.99

Table 6.2: SIMULATED1000 data set. CH index for the entropy subset and the full projected set of genes and ARI. The size of the subset m is varied, whence only 100 entropy selected genes suffice to attain an almost perfect agreement (ARI = 0.99) with respect to the true partition.

Similarly to the simulated data case, we examine the effect of the subset size on the YEASTG microarray data set. Again, values for the CH index in both the entropy selected genes and the full set of projected genes are reported along with the ARI scores. Results are summarized on Table 6.3.

m	CH subset	CH projected	ARI
50	58	1.5E+02	0.14
100	2.1E+03	1.5E+04	0.92
150	3E+03	4.6E+03	0.92
200	1.7E+04	1.1E+04	0.92

Table 6.3: YEASTG microarray data set. CH index for the entropy subset and the full projected set of genes and ARI. The size of the subset m is varied, whence 100 entropy selected genes suffice to attain an adjusted rand index ARI = 0.92 with respect to the true partition.

Comparison with other clustering algorithms [120] are reported in Table 6.4. Note, however, that we employed only half of the genes to construct the clustering model.

	Simulated (1000, 100, 4)	Yeast galactose (205, 20,4)
proposed	0.99 (4)	0.92 (4)
CLIC	1 (4)	0.97 (4)
k-means	0.68 (4)	0.87 (4)
HPCluster	1 (4)	0.83 (4)
CRC	0.90 (6)	0.97 (4)
MCLUST	1 (4)	0.97 (4)
k-boost	0.72 (3)	0.95 (4)
CLICK	NA (1)	0.81 (2)

Table 6.4: Adjusted rand indexes of several clustering algorithms for the SIMULATED1000 data set and the YEASTG microarray data set.

In order to illustrate the final ordering of the genes and reveal the cluster structure, the affinity (kernel) matrices of the entropy selected genes and the full set of projected are shown in Figure 6.4. On the left hand side, every row represents each of the 100 entropy selected genes used to build the spectral clustering model. Genes assigned to each of the 4 clusters identified by spectral clustering are shown on the right hand side. We appreciate two prominent

clusters, whence particularly cluster (4) reveals a large homogeneous group of under-expressed genes.

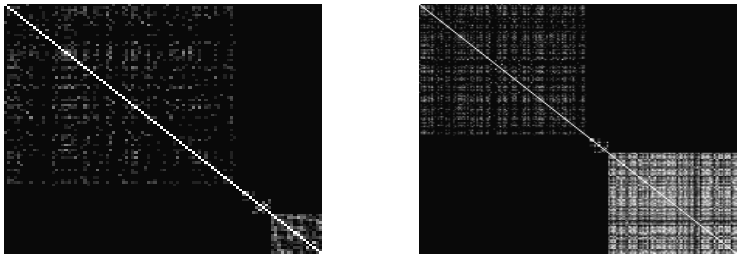


Figure 6.4: Affinity matrices revealing the clustering structure that is preserved both in the entropy selection phase and the extension to the full group of genes. **Left:** Entropy selected genes. **Right:** Full set of genes. The lower block represents the homogeneous cluster (4) of under-expressed genes.

6.3.4 The NCI60 data set

An example study is presented that uses NCI60 cell line data [88]. This large-scale data set consists of $n = 22283$ gene probes and $d = 108$ biological samples collected from various human cancer cell lines. As suggested by the authors, before clustering we centered in the gene direction as to make the subsequent analysis independent of the amount of each gene's mRNA in the reference pool. In a first attempt to cluster this large dataset, we let the subset size to take values $m \in \{100, 200, 500, 1000, 2000, 3000\}$, and construct a grid for the parameters $k \in [2, ..., 20]$, $\sigma \in [10^{-3}, ..., 500] \times \sqrt{d}$. Under these settings, we report in Figure 6.5, CH values averaged over 20 randomizations.

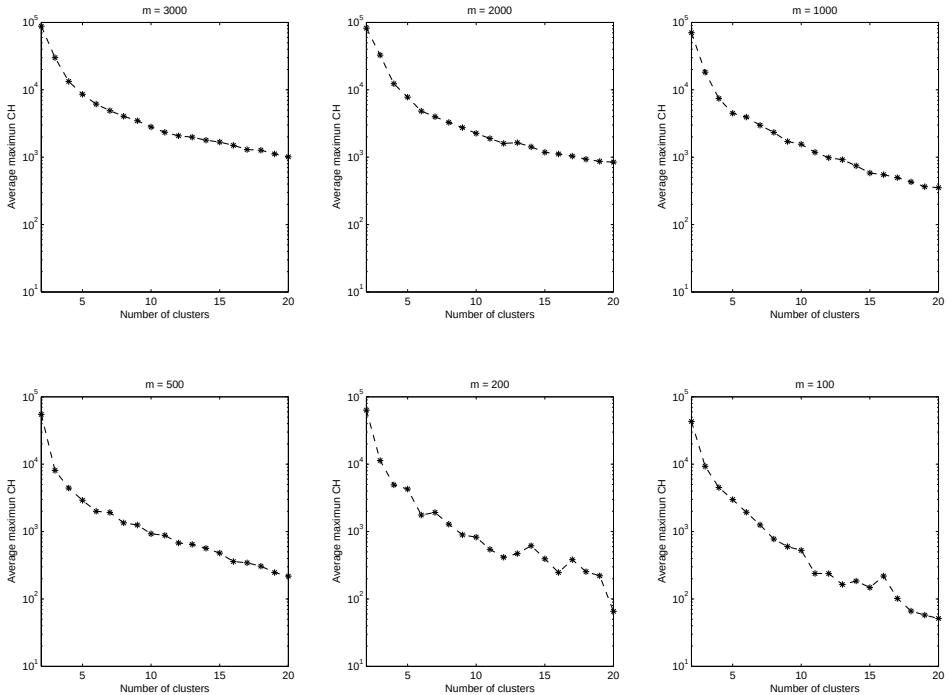


Figure 6.5: NCI60 microarray data set. We report the average CH index with respect to the number of clusters k and the subset size $m \in \{100, 200, 500, 1000, 2000, 3000\}$. In all cases, the CH index peaks at $k = 2$, hence suggesting the presence of large groups of genes organized in a hierarchical structure.

The monotonic decrease of the average CH values indicates the presence of two large prominent groups of genes. In turn, these results suggest that a global clustering approach might not be enough to reveal other smaller clusters. Moreover, given the relatively noisy character of microarray expression data, the more subtle clusters are likely to be indistinguishable from the noise. To this end, we adapt our algorithm to work in a hierarchical divisive fashion, that is, starting from the full set of genes we recursively partition it into smaller groups forming a tree. At each node of this tree, we construct first a subset using the entropy method, tune the parameters and finally cluster the remaining genes. A maximum depth of 10 levels is fixed and we restrict our search to a maximum of number of clusters $k_{\max} = 5$. Since clustering of the data occurs at different levels, and thus different resolutions, we reduce the range for σ at each level. The size of the subset is fixed to half of the data points at each level with $m_{\min} = 100$ and $m_{\max} = 1000$ as extreme values. Again, the best pair (k, σ) is selected according to the average maximum CH over 20 randomizations.

Our procedure delivers a hierarchical tree with 40 leaves. At each level the average maximum CH was once more attained at $k = 2$, hence we end up having a binary tree. Figure 6.6 displays the tree organization in the form of a dendrogram. Leave nodes indicate each of the 40 clusters found by our method, with the top-most node representing the full data set. The lower panel illustrates the distribution of the genes into the 40 clusters. We can readily appreciate three big clusters accounting for about 33% of the data set.

Further, we narrow down our analysis to those cluster groups containing less than 200 genes and perform a standard functional enrichment analysis using the web-tools DAVID[3] [48] and Babelomics[4] [4, 5]. The most statistically significant terms involving biological processes are listed in Tables 6.5 and 6.6. Notice that the clusters analysis reveals meaningful GO annotation associations and, in general, they tend to contain genes which represent specific biological processes. For example, the genes which are members of cluster 38 are involved in immune response mediated by leukocyte activation. Likewise, the members of cluster 23 are also involved in immune response, however, their mode of action is more specific to antigen mediated immune response, as is evident from inspection of the terms presented in Table 6.5. Cluster 19 contains genes

[3] `http://david.abcc.ncifcrf.gov/`
[4] `http://babelomics.bioinfo.cipf.es/`

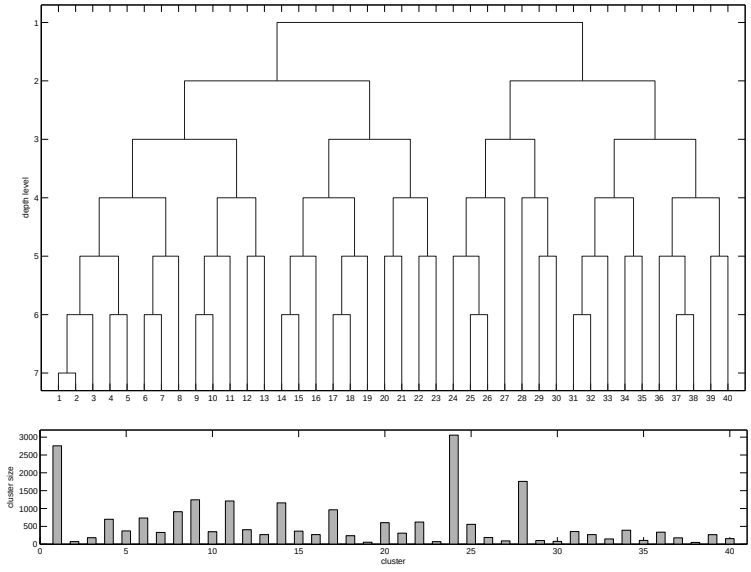


Figure 6.6: NCI60 data set. **Top.** Dendrogram representation of our hierarchical approach. At each level the optimal number of partitions was found to be $k = 2$. The result is then a binary tree with a total of 40 clusters on the leaves. **Bottom.** Distribution of the genes across the 40 clusters. Three major clusters can be easily distinguished as they account for about 33% of the data set.

which are associated almost exclusively with neurological processes such as *neuron differentiation* and *cell communication*. Cluster 30 is composed of genes which are predominately involved in the cell cycle and its regulation. The genes in Cluster 27 are involved in g-protein coupled signaling, whereas cluster 35 clearly represents genes exclusively associated to metabolism and catabolic processes.

Finally, we look at the terms in cluster 38 involving immune response and leukocyte activation, and explore how they relate to each other in the gene ontology annotation. The corresponding tree structure is presented in Figure 6.7.

Cluster (size)	Term (p-value)
cluster 38 (50)	lymphocyte activation (2.2E-4), B cell differentiation (5.9E-3), B cell activation (5.9E-3), T cell activation (5.9E-3), germ cell development (5.9E-3), elevation of cytosolic calcium ion concentration (8.5E-3), cytosolic calcium ion homeostasis (9.9E-5), ameboidal cell migration (9.9E-3).
cluster 19 (55)	negative regulation of neuron differentiation (5.4E-3), synaptic transmission (6.7E-3), multicellular organismal process (7.7E-3), cell-cell signaling (9.7E-3), cell communication (1.0E-2), localization (1.1E-2), transmission of nerve impulse (1.2E-2), neurological system process (1.2E-2), anatomical structure morphogenesis (1.4E-2), ion transport (1.5E-2).
cluster 23 (71)	antigen processing and presentation of peptide antigen via MHC class I (3.5E-3), antigen processing and presentation of peptide antigen (9.8E-3), antigen processing and presentation of endogenous peptide antigen via MHC class I (3.3E-2), antigen processing and presentation of endogenous peptide antigen (3.3E-2), antigen processing and presentation (4.0E-2), metabolic process (4.0E-2), primary metabolic process (4.3E-2), regulation of cellular metabolic process (4.3E-2), antigen processing and presentation of endogenous antigen (4.4E-2), regulation of primary metabolic process (4.5E-2).
cluster 30 (78)	regulation of cell cycle process (1.9E-2), signal transduction (2.1E-2), cell-cell signaling (2.7E-2), cell surface receptor linked signal transduction (3.1E-2), cell communication (3.1E-2), maintenance of protein location in nucleus (3.7E-2), regulation of mitotic cell cycle (3.8E-2), intracellular signaling cascade (5.8E-2), regulation of cell cycle (7.9E-2), protein localization in nucleus (8.0E-2).

Table 6.5: NCI60 data set. Most significant functional terms associated to each the clusters. Corresponding p-values are indicated between brackets.

Cluster (size)	Term (p-value)
cluster 27 (93)	G-protein coupled receptor protein signaling pathway (1.8E-3), cell surface receptor linked signal transduction (3.6E-3), second-messenger-mediated signaling (3.7E-3), secretion (8.1E-3), behavior (2.4E-2), cation homeostasis (3.1E-2), cellular metal ion homeostasis (3.6E-2), G-protein signaling, coupled to cyclic nucleotide second messenger (3.8E-2), immune response (3.9E-2), metal ion homeostasis (4.2E-2).
cluster 35 (106)	aminoglycan catabolic process (6.0E-3), polysaccharide catabolic process (1.2E-2), carbohydrate catabolic process (3.8E-2), translational elongation (3.8E-2), chitin metabolic process (4.6E-2), detection of bacterium (4.6E-2), chitin catabolic process (4.6E-2), translation (5.6E-2), aminoglycan metabolic process (5.7E-2), immune response (7.9E-2)
cluster 33 (148)	behavior (7.5E-4), monocarboxylic acid metabolic process (1.0E-2), regulation of synaptic transmission (1.2E-2), associative learning (1.5E-2), cellular homeostasis (1.5E-2), homeostatic process (1.6E-2), regulation of secretion (1.6E-2), regulation of transmission of nerve impulse (1.7E-2), carbohydrate metabolic process (1.9E-2), regulation of neurological system process (2.1E-2)
cluster 40 (155)	nucleobase, nucleoside, nucleotide and nucleic acid metabolic process (8.7E-17), cellular nitrogen compound metabolic process (1.3E-14), nitrogen compound metabolic process (7.2E-14), cellular macromolecule metabolic process (4.8E-13), macromolecule metabolic process (1.1E-10), cellular metabolic process (5.5E-9), cellular macromolecule biosynthetic process (3.1E-8), primary metabolic process (3.8E-8), macromolecule biosynthetic process (4.0E-8), DNA metabolic process (8.4E-8)

Table 6.6: NCI60 data set. Most significant functional terms associated to each the clusters. Corresponding p-values are indicated between brackets.

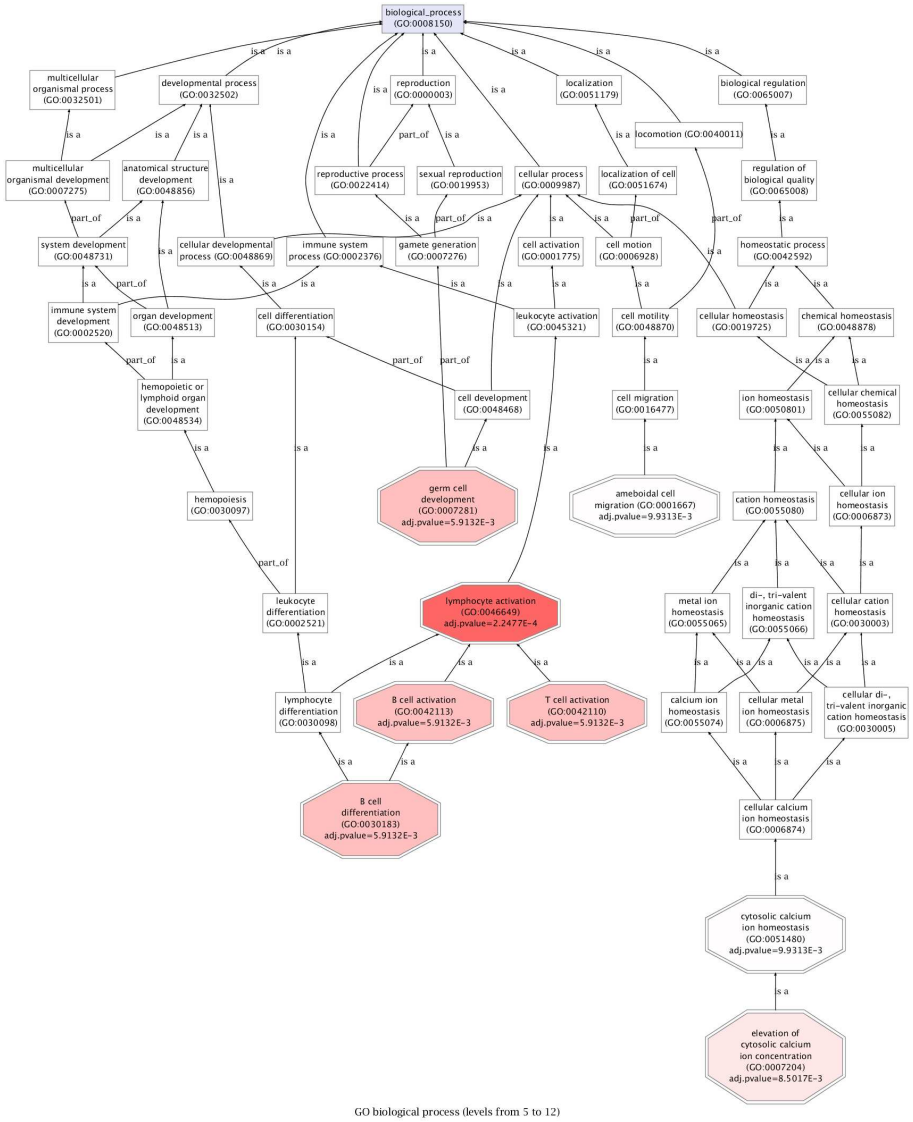


Figure 6.7: NCI60 data set. Gene ontology tree structure for the most significant terms of the genes in cluster 38.

6.4 Summary

With the ongoing availability of large microarray data sets, clustering algorithms should move also forward to reduce large computational complexity and memory requirements. This study presents a modified spectral clustering approach, which handles large data sets and it is not limited to microarray data sets. Our method is based on a novel concept in which informative genes (or data points) are first selected using an entropy criterion. This selection step leads to a more compact and better representation of the full data set than a random selection. The entropy genes, in turn, form the support set upon which the spectral clustering model is developed. Moreover, in order to detect subtle clusters in large scale data sets and therefore avoid forming oversimplified clusters, we adapt our methodology to work in a hierarchical fashion with clusters being refined at each level.

Chapter 7

General Conclusions

7.1 Concluding Remarks

In this thesis we have discussed a series of methodologies for learning regularized models and kernel based methods, mostly based on the least squares support vector machine (LS-SVM) algorithm. Core LS-SVM models in supervised learning are extended towards variable selection problems and adapted to exploit the structure of the algorithms as well as incorporate prior problem information. In particular the presented methodologies are aimed for data sets with few number of data points and large number of variables, which often appear in the context of microarray and mass spectrometry data analysis. We explore the applicability of the proposed methodologies through a range of problems in classification, variable selection, prediction and clustering, always giving special attention to issues in model selection and algorithm efficiency. The work presented the following topics:

- **Low rank-updates for linear kernels**. Application of *low rank updates/downdates* to the structure (linear kernel) of the LS-SVM classifiers was found to be very suitable for variable selection wrapper-like algorithms. By exploiting the inherent block structure of the LS-SVM solution, an efficient method for computing the leave-one-out enabled us to evaluate the performance of potentially relevant subsets of variables. Furthermore, in the special case of the *linear kernel*, algorithms for fast

forward and *backward search* are devised such that *matrix inversions* are at most completely avoided.

- **Polynomial componentwise LS-SVM**. The core learning algorithm is the (componentwise) LS-SVM classifier that can be efficiently trained by solving just one system of linear equations. The main concept presented, however concerns the the use of a valid explicit *feature map* for polynomial kernels in an additive construction. By exploiting the structure of the feature map, it is demonstrated how the model parameters of the classification/regression problem can be easily modified and updated when new variables enter the current model, while still providing non-linear modeling capabilities. This is achieved by the use of low rank updates which in turn constitutes an algorithmic tool for the design of sequential variable ranking algorithms in high dimensional settings. Moreover relevant variables can be robustly ranked using the closed form of the leave-one-out (LOO) error estimator directly obtained as a by-product of the low rank modifications.

- **Efficient model selection based on LOO estimator**. We extend the use of the efficient leave-one-out formulation of LS-SVM classifiers to the domain of variable selection. In combination with low rank updates we arrived at a very fast and robust methodology to rank important variables.

- **Sparse linear models for structured data**. Starting from regularized learning models and structural information inherent to MSI data, we make use of the graph Laplacian to embed first, the natural ordering of the m/z variables and, secondly the spatial location of the spectra. Thereby, smooth quadratic penalties are imposed over neighboring *nodes* representing in the first case *variables* and in the second one *data points*. These penalties modify the standard learning algorithm resulting in an equivalent *lasso* formulation that can be solved efficiently.

- **Unlabeled data**. In order to cope with the lack of labeled data, typical to MSI experiments, we proposed to model the predicted responses via the graph Laplacian. The applicability of the proposed approach is explored in a mouse brain MSI data set to distinguish amongst four anatomical regions, and it is compared to other learning models that do not, or partially, incorporate the structural information of MSI data.

- **Entropy selection for spectral clustering**. For clustering of microarray data involving thousands of genes, e.g. 22000, we have introduced a subset selection mechanism based on entropy selection. Such subsampling not only provides informative genes best representing the underlying data set, but it also makes the spectral clustering tractable for large scale problems. Additionally, we have shown how to perform spectral clustering in a hierarchical fashion to unravel subtle clusters that otherwise are difficult to detect using a global partitioning approach.

7.2 Future research and open issues

The work presented in this thesis drifts gradually from a theoretical perspective towards practical examples and applications on real-life data. Nevertheless, it is important to highlight some open issues and future directions.

- Through this thesis we have explored three different learning paradigms: (i) supervised classification using LS-SVM, (ii) unsupervised learning with spectral clustering and (iii) semi-supervised learning through sparse linear models. The latter represents best scenarios in practice since prior knowledge is often available. Incorporation of such information into the learning models covers a major direction for future extensions. As we have seen on the MSI application, the structural information is introduced in the form of constraints or extra penalty terms. In the case of clustering of genes, for instance, Gene Ontology (GO) databases would help increase the correlation between the learning problem at hand, the measured experiments and the background knowledge collected over many years. Fully unsupervised modeling in bioinformatics is perhaps overly optimistic.
- Successful application of a kernel method and/or regularized models depends greatly on appropriate kernel functions and correct tuning of associated hyper-parameters. We have shown how the fast leave-one-out criteria for LS-SVM based model not only provides a statistically sounded methodology but also enables efficiently algorithmic short-cuts. Nevertheless, resampling techniques such as bootstrapping and boosting could enhance robustness and deliver confidence intervals. Additionally,

the derivation of tighter bounds on the generalization error could initially provide a fast baseline approach.

- Extensions of the presented spectral clustering algorithm to more general hierarchical models. For example, combination of top-down and bottom-up schemes as a refinement step for the discovered clusters. Additionally, the use of gene ontology could serve as a constraint set during the refinement phase or even help determine the number of clustering levels. This way, one might state the problem in a semi-supervised setting.
- Another direction for future research consists in extending many of the presented algorithms to further incorporate structural information and prior knowledge coming from heterogeneous sources of information. If available, putting together genomic information from, for instance microarray data, with mass spectrometry data sets should provide a dual view at the gene and protein levels.
- Unfortunately, the lack of publicly available data sets in Mass Spectral Imaging (MSI) represents a major obstacle for the development and, more importantly, the validation of newer methodologies and learning algorithms. Additionally, this absence also prevents from further spread of recent advances in data mining into the MS/MSI communities. Therefore, standardization and available repositories will not only boost up new MS/MSI applications, but it will also bridge the gap among different research communities.

Appendix A

Benchmark results for linear low ranked LS-SVM

This appendix presents detailed results of the benchmarking study which complement those earlier exposed in Chapter 3.

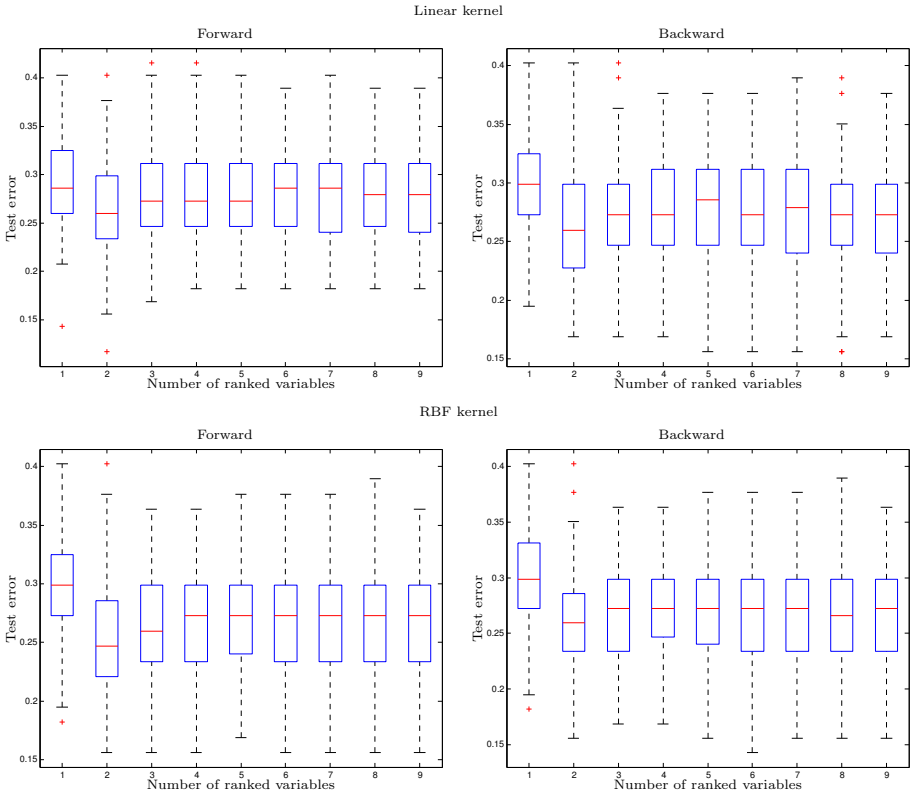


Figure A.1: BREAST CANCER

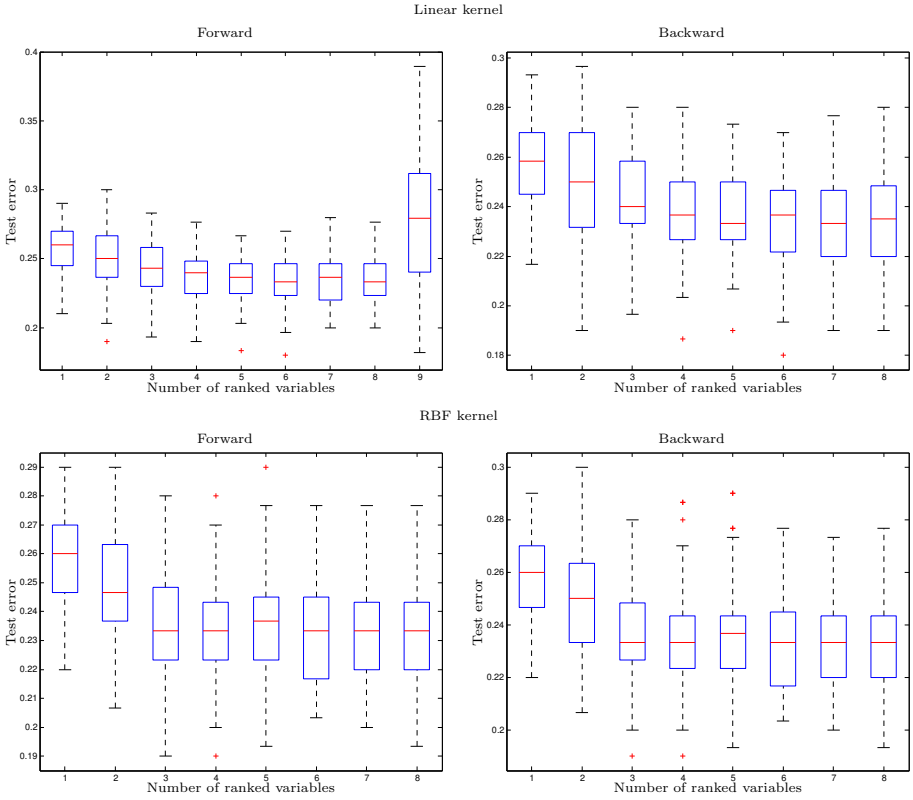


Figure A.2: DIABETES

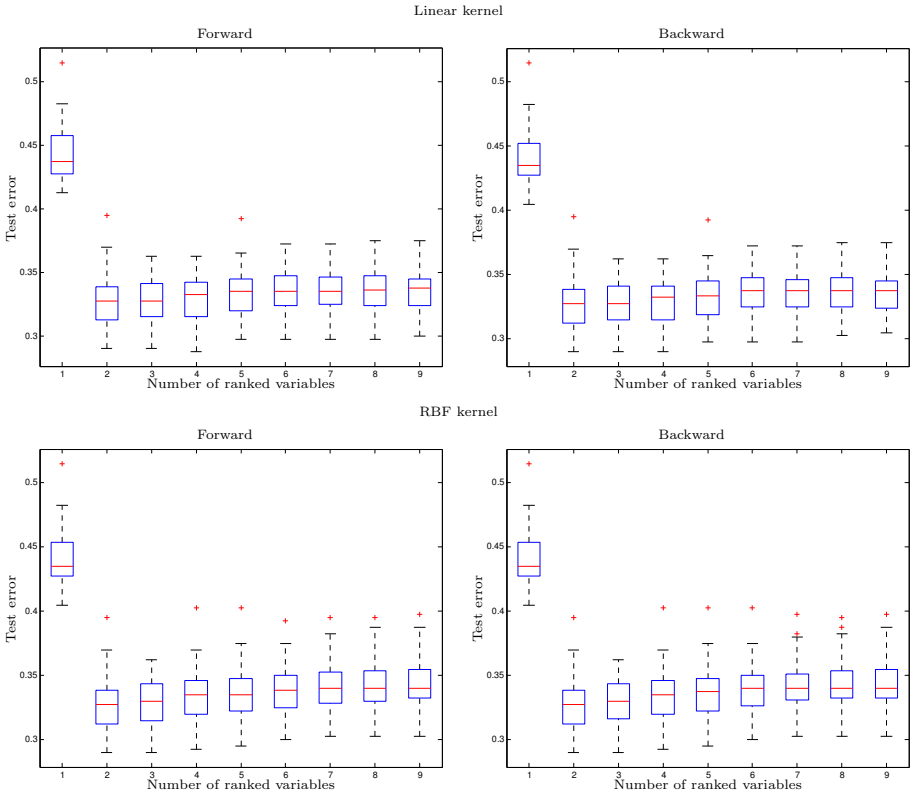


Figure A.3: FLARE SOLAR

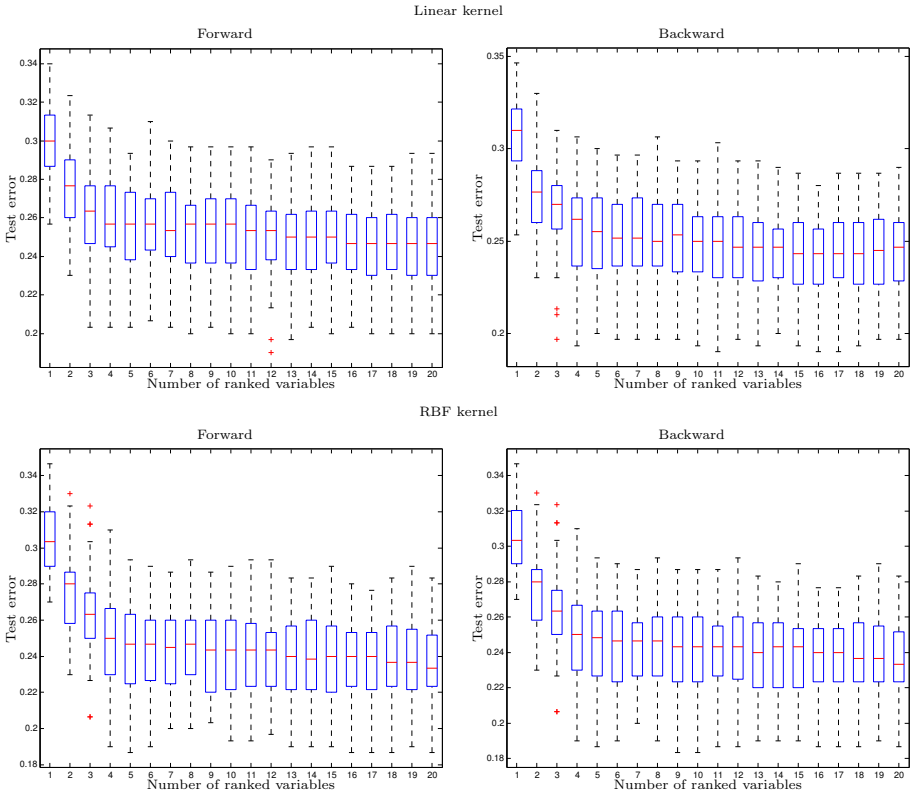


Figure A.4: GERMAN

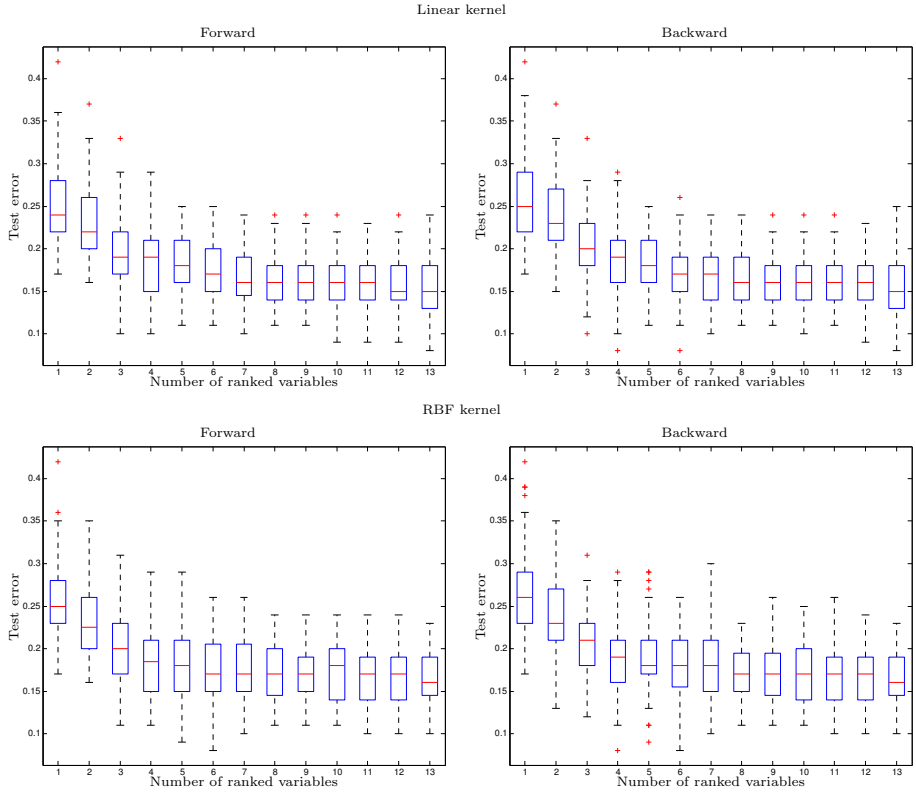


Figure A.5: HEART

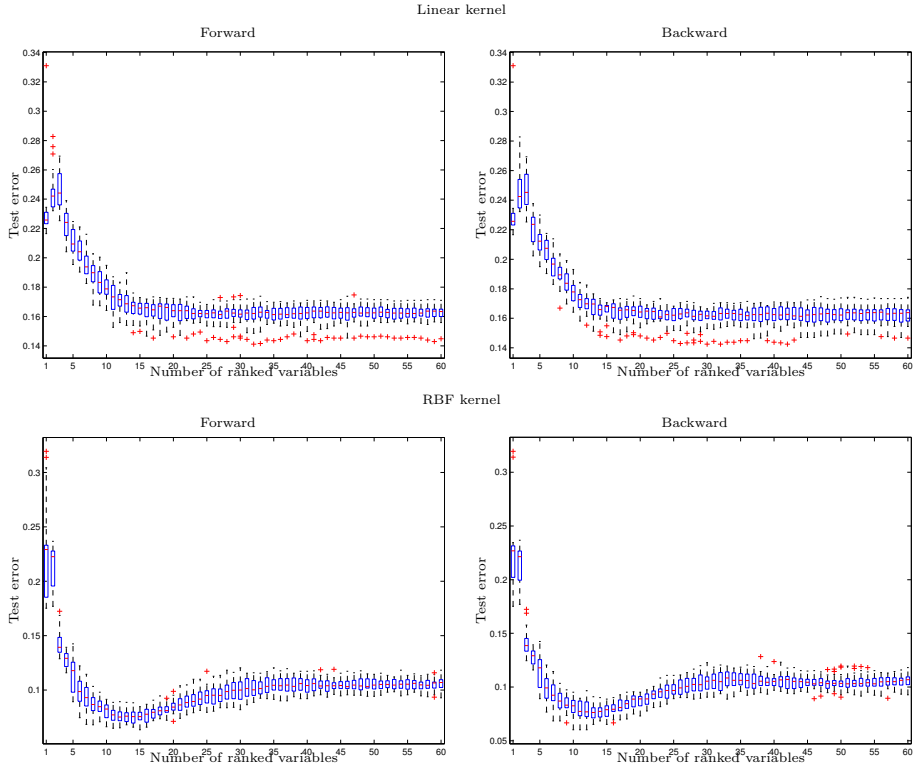


Figure A.6: SPLICE

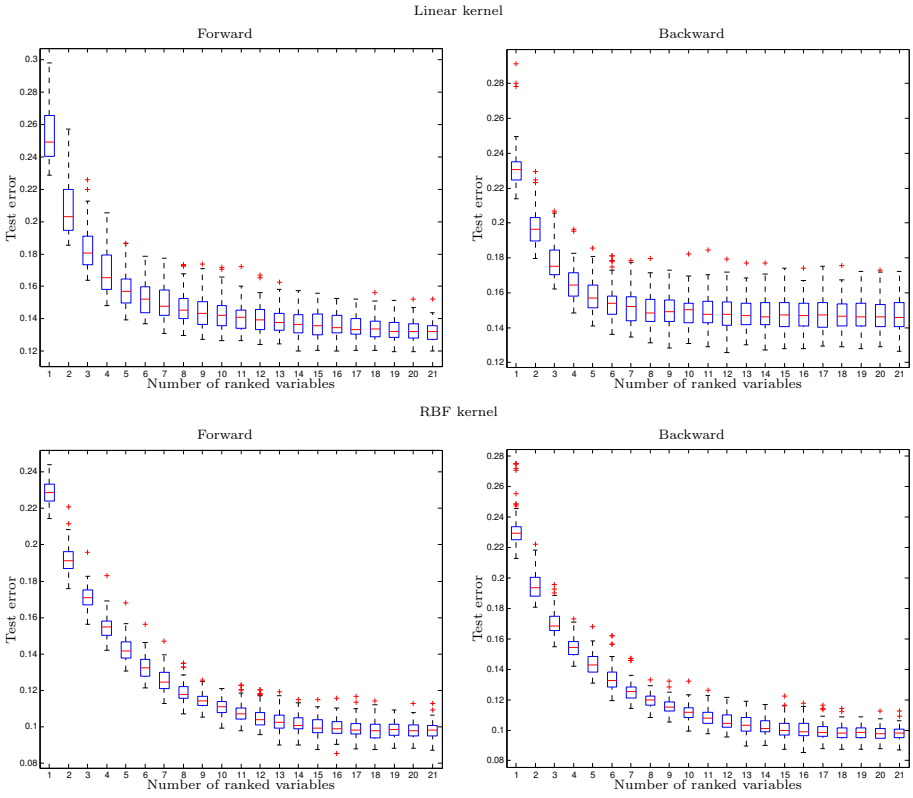
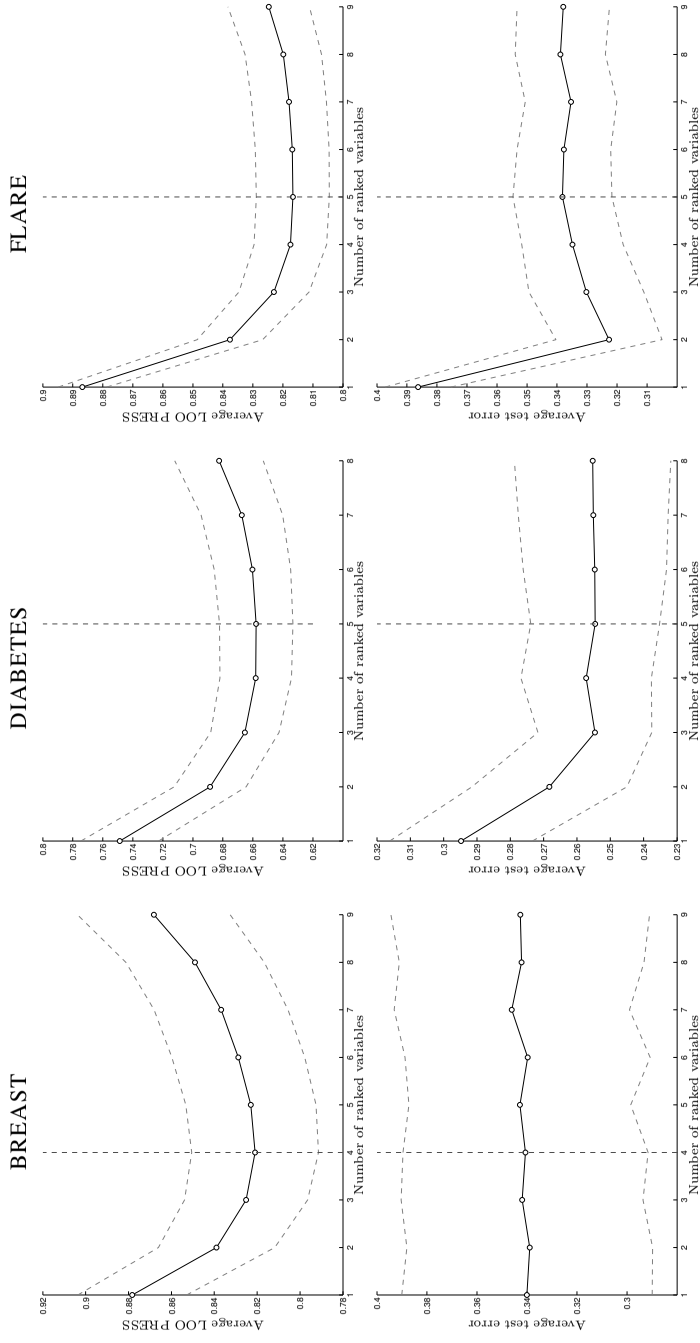


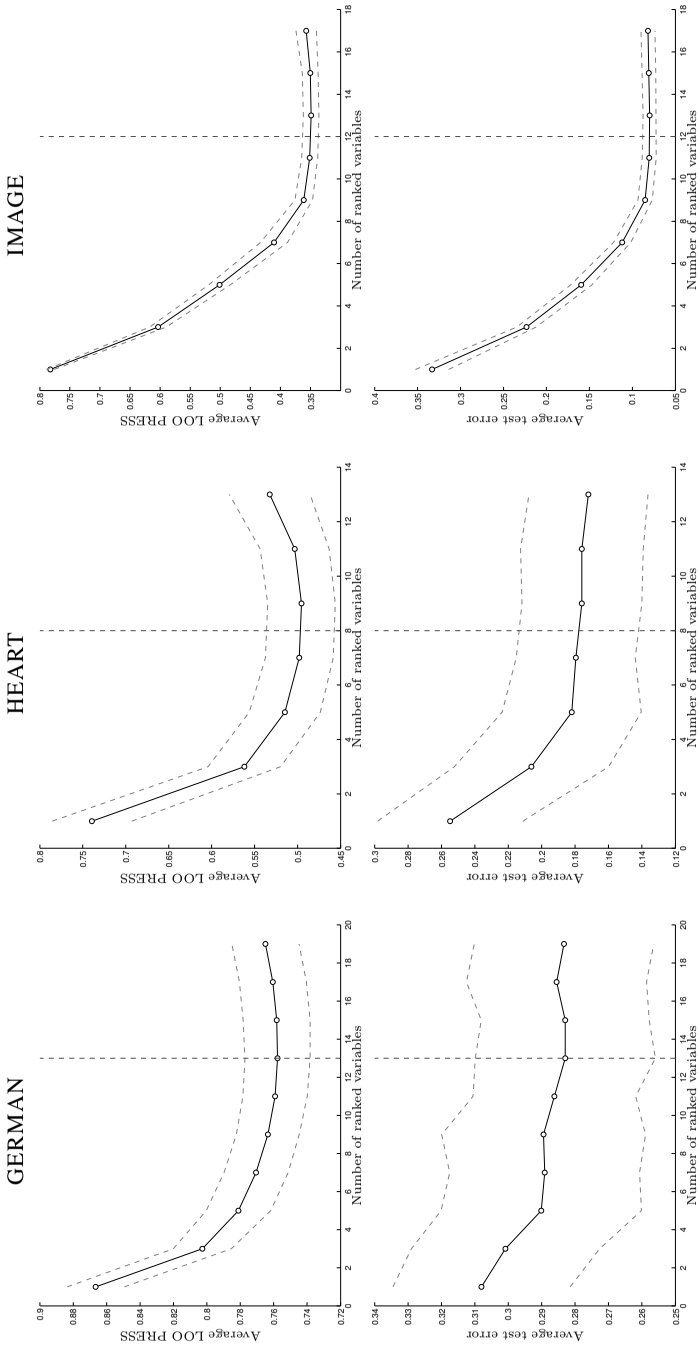
Figure A.7: WAVEFORM

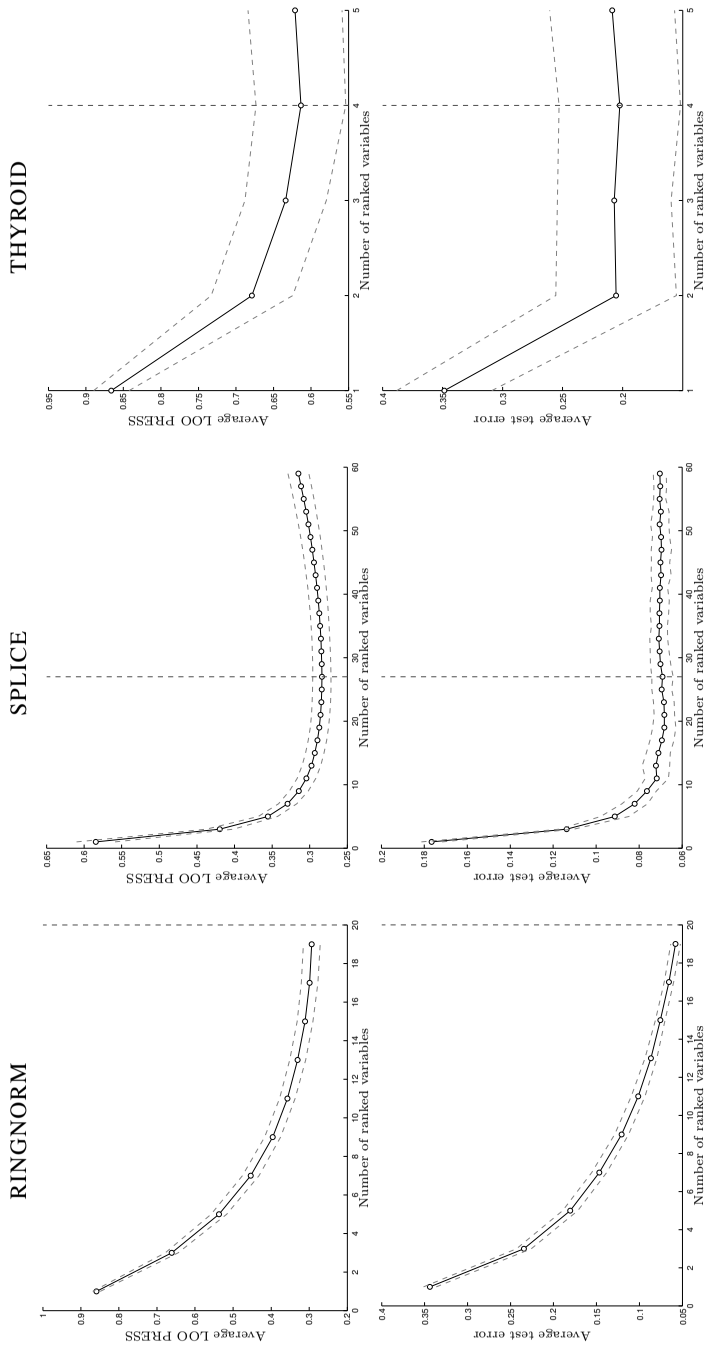
Appendix B

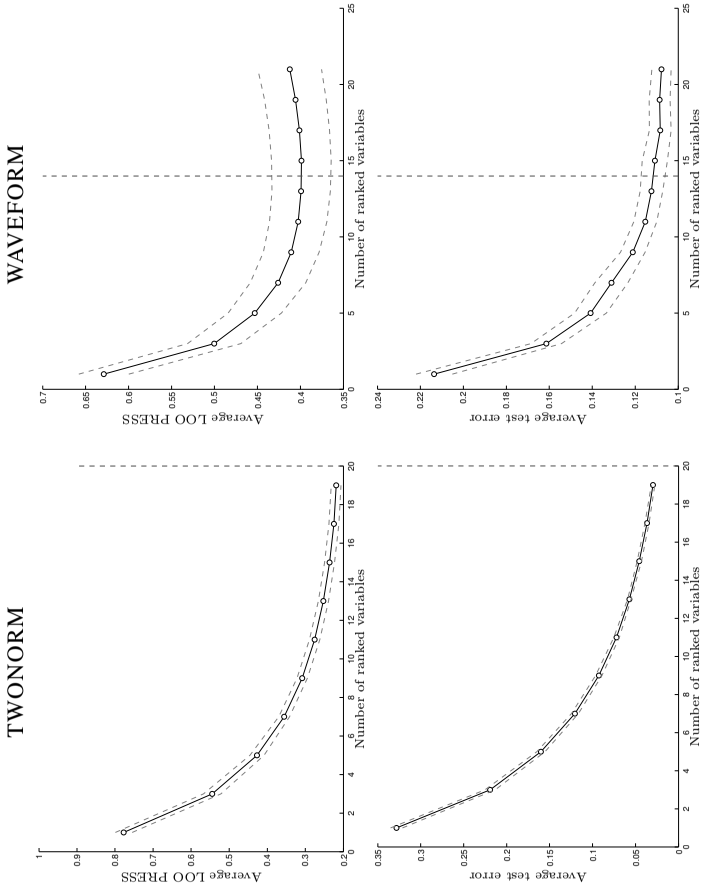
Benchmark results for polynomial componentwise LS-SVM

This appendix presents detailed results of the benchmarking study which complement those earlier exposed in Chapter 4.


FLARE
DIABETES
BREAST
Average LOO PRESS
Average test error
Number of ranked variables


GERMAN
HEART
IMAGE
Average LOO PRESS
Average test error
Number of ranked variables


RINGNORM
SPLICE
THYROID
Average LOO PRESS
Average test error
Number of ranked variables


WAVEFORM
TWONORM
Average LOO PRESS
Average test error
Number of ranked variables

Bibliography

[1] FDA-NCI Clinical Proteomics Program Databank. National Cancer Institute, 2002. pages 27

[2] M.A. Aizerman, E.M. Braverman, and L. Rozonoèr. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25(6):821–837, 1964. pages 17

[3] A.L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997. pages 24

[4] F. Al-Shahrour, J. Carbonell, P. Minguez, S. Goetz, A. Conesa, J. Tárraga, I. Medina, E. Alloza, D. Montaner, and J. Dopazo. BABELOMICS: advanced functional profiling of transcriptomics, proteomics and genomics experiments. *Nucleic Acids Res*, 36(Web Server issue):W341–6, July 2008. pages 107

[5] F. Al-Shahrour, P. Minguez, J.M. Vaquerizas, L. Conde, and J. Dopazo. BABELOMICS: a suite of web tools for functional annotation and analysis of groups of genes in high-throughput experiments. *Nucleic Acids Res*, 33(Web Server issue):W460–4, July 2005. pages 107

[6] D.M. Allen. The relationship between variable selection and data agumentation and a method for prediction. *Technometrics*, 16(1):pp. 125–127, 1974. pages 26

[7] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc Natl Acad Sci*, 96(12):6745–6750, 1999. pages 47

[8] C. Alzate and J.A.K. Suykens. A weighted kernel PCA formulation with out-of-sample extensions for spectral clustering method. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'06)*, pages 138–144, 2006. pages 8, 89, 90, 93

[9] C. Alzate and J.A.K. Suykens. Multiway spectral clustering with out-of-sample extensions through weighted kernel pca. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(2):335–347, 2010. pages 7, 8, 89, 90, 93

[10] C. Ambroise and G. J. McLachaln. Selection bias in gene extraction on the basis of microarray gene-expression data. *PNAS*, 99(10):6562–6566, 2002. pages 54

[11] S. An, W. Liu, and S. Venkatesh. Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. *Pattern Recognition*, 40(8):2154–2162, 2007. pages 31

[12] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK user's guide*. Software - Environments - Tools. 9. Philadelphia, PA: SIAM, Society for Industrial and Applied Mathematics. xxi, 407 p., 1999. pages 41

[13] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, May 1950. pages 16

[14] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, November 2006. pages 80

[15] S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge Univ Pr, 2004. pages 18

[16] M.P.S. Brown, W. Noble, D. Lin, N. Cristianini, C.W. Sugnet, T.S. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1):262–267, 2000. pages 3, 29, 99

[17] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998. pages 63

[18] R.B. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3:1–27, 1974. pages 97

[19] G.C. Cawley. Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1661–1668, Vancouver, 2006. pages 9, 26, 31, 35, 36, 37, 66, 70

[20] G.C. Cawley and N.L.C. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17:1467–1475, 2004. pages 9, 35

[21] G.C. Cawley and N.L.C. Talbot. Preventing over-fitting in model selection via bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861, 2007. pages 56, 70

[22] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002. pages 25, 70

[23] F.R.K. Chung. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society, February 1997. pages 79, 90, 92, 93

[24] G. Claeskens, C. Croux, and J. Van Kerckhoven. An information criterion for variable selection in support vector machines. *J. Mach. Learn. Res.*, 9:541–558, 2008. pages 70

[25] K. De Brabanter, J. De Brabanter, J.A.K. Suykens, and B. De Moor. Optimized fixed-size kernel models for large data sets. *Comput. Stat. Data Anal.*, 54:1484–1504, June 2010. pages 96, 97

[26] I.S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, New York, NY, USA, 2004. ACM Press. pages 90

[27] J.J. Dongarra, J.R. Bunch, C.B. Moler, and G.W. Stewart. LINPACK Users' Guide. *SIAM, Philadelphia*, 1979. pages 39, 40

[28] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–451, 2004. pages 16, 77, 81

[29] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, pages 147–153, 2003. pages 97

[30] M. Espinoza, J.A.K. Suykens, and B. De Moor. Fixed-size least squares support vector machines: A large scale application in electrical load forecasting. *Computational Management Science*, 3(2):113–129, April 2006. pages 97

[31] B.S. Everitt, E. Landau, and M. Leese. *Cluster Analysis*. Arnold, London, 2001. pages 98

[32] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001. pages 39

[33] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 2004, 2004. pages 93

[34] P. Fränti and O. Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recogn.*, 39:761–775, May 2006. pages 99

[35] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007. pages 16

[36] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Storz, D. Botstein, and P.O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11(12):4241–4257, 2000. pages 29

[37] P.E. Gill, G.H. Golub, W. Murray, and M.A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28(126):505–535, Apr. 1974. pages 39, 41

[38] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, 1981. pages 64, 65, 66

[39] G.H. Golub and C.F. Van Loan. *Matrix computations*. The Jonhs Hopkins University Press, 2nd. edition, 1989. pages 23, 35, 38, 39, 40, 41, 64, 65

[40] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002. pages 38, 49, 50, 54, 55, 74

[41] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2):107–145, 2001. pages 98

[42] M. Hanselmann, U. Köthe, M. Kirchner, B.Y. Renard, E.R. Amstalden, K. Glunde, R.M.A. Heeren, and F.A. Hamprecht. Toward digital staining using imaging mass spectrometry and random forests. *Journal of Proteome Research*, 8(7):3558–3567, 2009. pages 77

[43] C.T. Harbison, D.B. Gordon, T.I. Lee, N.J. Rinaldi, K.D. Macisaac, T.W. Danford, N.M. Hannett, J. Tagne, D.B. Reynolds, J. Yoo, E.G. Jennings, J. Zeitlinger, D.K. Pokholok, M. Kellis, P.A. Rolfe, K.T. Takusagawa, E.S. Lander, D.K. Gifford, E. Fraenkel, and R. A. Young. Transcriptional regulatory code of a eukaryotic genome. *Nature*, 431(7004):99–104, 09 2004. pages 30

[44] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall, New York, 1990. pages 60, 68

[45] D.T. Holloway, M. Kon, and C. DeLisi. Machine learning for regulatory analysis and transcription factor target prediction in yeast. *Systems and Synthetic Biology*, 1(1):25–46, 2007. pages 30, 31

[46] X. Hong, S. Chen, and P.M. Sharkey. Automatic kernel regression modelling using combined leave-one-out test score and regularised orthogonal least squares. *International Journal Neural Systems*, 14(1):27–37, February 2004. pages 43

[47] Z. Hu, J. Mellor, J. Wu, and C. DeLisi. VisANT: an online visualization and analysis tool for biological interaction data. *BMC Bioinformatics*, 5(1):17, 2004. pages 31

[48] D.W. Huang, B.T. Sherman, and R.A. Lempicki. Systematic and integrative analysis of large gene lists using david bioinformatics resources. *Nat Protoc*, 4(1):44–57, 2009. pages 107

[49] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. pages 101

[50] T. Ideker, V. Thorsson, J.A. Ranish, R. Christmas, J. Buhler, J.K. Eng, R. Bumgarner, D.R. Goodlett, R. Aebersold, and L. Hood. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science*, 292(5518):929–34, May 2001. pages 103

[51] J.A. Hanley and B.J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982. pages 28

[52] J.C. Platt. *Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in P.J. Bartlett, B. Schölkopf, D. Schuurmans and A.J. Smola (Eds.), Advances in large margin classifiers*. MIT Press, Cambridge, 2000. pages 31

[53] J. Kim, Y. Kim, and Y. Kim. A gradient-based optimization algorithm for lasso. *Journal of Computational and Graphical Statistics*, 17(4):994–1009, 2008. pages 16

[54] R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997. pages 24, 25, 34

[55] J.T. Kwok and I.W. Tsang. The pre-image problem in kernel methods. *IEEE Trans Neural Networks*, 15(6):1517–25, November 2004. pages 92

[56] P.A. Lachenbruch. An almost unbiased method for the probability of misclassification in discriminant analysis. *Biometrics*, 23:639–645, 1967. pages 26

[57] K. Lemmens, T. Dhollander, T. De Bie, P. Monsieurs, K. Engelen, B. Smets, J. Winderickx, B. De Moor, and K. Marchal. Inferring transcriptional modules from ChIP-chip, motif and microarray data. *Genome Biology*, 7(5):R37, 2006. pages 30

[58] C. Li and H. Li. Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, 24(9):1175–1182, 2008. pages 80, 82

[59] J. Luts, F. Ojeda, R. Van de Plas, B. De Moor, S. Van Huffel, and J.A.K. Suykens. A tutorial on support vector machine-based methods

for classification problems in chemometrics. *Analytica Chimica Acta*, 665(2):129–145, 2010. pages 77

[60] D.J.C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4:720–736, 1992. pages 60

[61] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010. pages 41

[62] G. McCombie, D. Staab, M. Stoeckli, and R. Knochenmuss. Spatial and spectral correlations in MALDI mass spectrometry images by clustering and multivariate analysis. *Analytical Chemistry*, (19):6118–6124, 2005. pages 77

[63] L.A. McDonnell and R.M.A. Heeren. Imaging mass spectrometry. *Mass Spectrometry Reviews*, 26(4):606–643, 2007. pages 76

[64] M. Meila and J. Shi. A random walks view of spectral segmentation. In *AISTATS*, 2001. pages 94

[65] D.J. Michaud, A.G. Marsh, and P.S. Dhurjati. eXPatGen: generating dynamic expression patterns for the systematic evaluation of analytical methods. *Bioinformatics*, 19(9):1140–6, June 2003. pages 103

[66] G.W. Milligan and M.C. Cooper. An examination of procedures of determining the number of cluster in a data set. *Psychometrika*, 50(2):159–179, 1985. pages 97

[67] F. Mordelet and J.P. Vert. SIRENE: supervised inference of regulatory networks. *Bioinformatics*, 24(16):i76–i82, 2008. pages 31

[68] I. T. Nabney. Efficient training of rbf networks for classification. *Int. J. Neural Syst.*, 14(3):201–208, 2004. pages 97

[69] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *In Advances in Neural Information Processing Systems*, volume 14, 2001. pages 90, 98

[70] F. Ojeda, C. Alzate, B. De Moor, and J.A.K Suykens. Entropy based selection for spectral clustering. Technical Report 11-83, ESAT-SISTA K.U. Leuven., 2011. pages 11

[71] F. Ojeda, C. Alzate, J.A.K. Suykens, and Bart De Moor. A large scale spectral clustering with out of sample extension: application to gene expression data. 15th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB) and 6th European Conference on Computational Biology (ECCB), pages 1–1, July 2007. Outstanding poster award. pages 11

[72] F. Ojeda, T. Falck, B. De Moor, and J.A.K Suykens. Polynomial componentwise LS-SVM: fast variable selection using low rank updates. In *International Joint Conference on Neural Networks (IJCNN 2010)*, pages 3291–3297, July 2010. pages 9

[73] F. Ojeda, M. Signoretto, R. Van de Plas, E. Waelkens, B. De Moor, and J.A.K Suykens. Semi-supervised learning of sparse linear models in mass spectral imaging. In T. Dijkstra, E. Tsivtsivadze, E. Marchiori, and T. Heskes, editors, *Pattern Recognition in Bioinformatics*, volume 6282 of *Lecture Notes in Computer Science*, pages 325–334. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-16001-1_28. pages 10

[74] F. Ojeda, J.A.K. Suykens, and B. De Moor. Variable selection by rank-one updates for least squares support vector machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'07)*, pages 2283–2288, Orlando, U.S.A., 2007. pages 9

[75] F. Ojeda, J.A.K. Suykens, and B. De Moor. Low rank updated LS-SVM classifiers for fast variable selection. *Neural Networks*, 21(2-3):437–449, 2008. pages 9, 60, 64, 69, 70

[76] M.J.L. Orr. Regularisation in the selection of radial basis function centres. *Neural Computation*, pages 606–623, 1995. pages 97

[77] M.R. Osborne, B. Presnell, and B.A. Turlach. On the LASSO and its dual. *Journal of Computational and Graphical statistics*, 9(2):319–337, 2000. pages 16

[78] T. Pahikkala, A. Airola, and T. Salakoski. Speeding up greedy forward selection for regularized least-squares. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pages 325–330, dec. 2010. pages 44

[79] K. Pelckmans, I. Goethals, J. De Brabanter, J.A.K. Suykens, and B. De Moor. *Componentwise least squares support vector machines*, chapter in Support Vector Machines: Theory and Applications, pages 77–98. Springer, 2005. pages 60, 61, 63

[80] K. Pelckmans, J.A.K. Suykens, and B. De Moor. Building sparse representations and structure determination on LS-SVM substrates. *Neurocomputing*, (64):137–159, 2005. pages 60, 62

[81] E. Petricoin III, A. Ardekani, B. Hitt, P. Levine, V. Fusaro, S. Steinberg, G.Mills, C. Simone, D. Fishman, and E. Kohn. Use of proteomic patterns in serum to identify ovarian cancer. *The Lancet*, 359(9306):572–577, 2002. pages 27, 28

[82] E. Petricoin III, D. Ornstein, C. Paweletz, A. Ardenaki, P. Hackett, B. Hitt, A. Velasco, C. Trucco, L. Wiegand, K. Wood, C. Simone, P. Levine, W. Marston, M. Linehan, M. Emmer-Buck, S. Steinberg, E. Kohn, and L. Liotta. Serum proteomic patterns for detection of prostate cancer. *Journal on the National Cancer Institute*, 94(20):1576–1578, 2002. pages 27, 28

[83] N. Pochet, F. Ojeda, F. De Smet, T. De Bie, J.A.K Suykens, and B. De Moor. *Kernel clustering for knowledge discovery in clinical microarray data analysis*, chapter III, pages 64–92. Kernel Methods in Bioengineering, Signal and Image Processing. Idea Group Inc., Hershey, Pennsylvania, 2007. pages 11, 90, 92, 98, 99

[84] J. Qian, J. Lin, N. M. Luscombe, H. Yu, and M. Gerstein. Prediction of regulatory networks: genome-wide identification of transcription factor targets from gene expression data. *Bioinformatics*, 19(15):1917–1926, 2003. pages 31

[85] A. Rakotomamonjy. Variable selection using SVM based criteria. *J. Mach. Learn. Res.*, 3:1357–1370, 2003. pages 70

[86] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for machine learning*. MIT Press, Cambridge, 2006. pages 25

[87] G. Rätsch, T. Onoda, and K.R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001. pages 44, 45, 66, 70

[88] D.T. Ross, U. Scherf, M.B. Eisen, C.M. Perou, C. Rees, P. Spellman, V. Iyer, S.S. Jeffrey, M. Van de Rijn, M. Waltham, A. Pergamenschikov, J.C. Lee, D. Lashkari, D. Shalon, T.G. Myers, J.N. Weinstein, D. Botstein, and P.O. Brown. Systematic variation in gene expression patterns in human cancer cell lines. *Nat Genet*, 24(3):227–35, March 2000. pages 105

[89] P. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20:53–65, November 1987. pages 99

[90] G. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proc. 15th International Conf. on Machine Learning*, pages 515–521. Morgan Kaufmann, San Francisco, CA, 1998. pages 23

[91] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.R. Müller, G. Rätsch, and A.J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10:1000–1017, 1999. pages 63

[92] B. Schölkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, 2001. pages 64

[93] M. Seeger. Low rank updates for the Cholesky decomposition. Technical Report. Max Planck Society, Tuebingen, Germany, 2005. pages 39, 40

[94] J. Sherman and W.J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):pp. 124–127, 1950. pages 39

[95] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. pages 90, 93, 94

[96] A. Shilton, M. Palaniswami, D. Ralph, and A. Chung. Incremental training of support vector machines. *IEEE Transactions on Neural Networks*, 16(1):114–131, 2005. pages 39

[97] M. Signoretto, A. Daemen, C. Savorgnan, and J.A.K. Suykens. Variable selection and grouping with multiple graph priors. In *2nd Neural*

Information Processing Systmes (NIPS) Workshop on Optimization for Machine Learning, 2009. pages 80, 82

[98] P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*, 9:3273–3297, 1998. pages 89

[99] M. Stoeckli, P. Chaurand, D.E. Hallahan, and R.M. Caprioli. Imaging mass spectrometry: A new technology for the analysis of protein expression in mammalian tissues. *Nature Medicine*, 7(4):493–496, April 2001. pages 76

[100] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002. pages 2, 21, 35, 93, 96, 97

[101] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, Jun 1999. pages 2, 21, 22, 34

[102] E.K. Tang, P.N. Suganthan, and X. Yao. Gene selection for microarray data based on least squares support vector machine. *BMC Bioinformatics*, 7(95), 2006. pages 48, 49, 54

[103] T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923, 1998. pages 24

[104] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996. pages 15

[105] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal of The Royal Statistical Society Series B*, 67(1):91–108, 2005. pages 77, 82

[106] V. Van Belle, K. Pelckmans J.A.K Suykens, and S. Van Huffel. Additive survival least squares support vector machines. *Statistics in Medicine*, (2):296–308, January 2010. pages 62

[107] R. Van de Plas, F. Ojeda, M. Dewil, L. Van Den Bosch, B. De Moor, and E. Waelkens. Prospective exploration of biochemical tissue composition via imaging mass spectrometry guided by principal component analysis. In *Proceedings of the Pacific Symposium on Biocomputing 12*, pages 458–469, Maui, 2007. pages 77

[108] R. Van de Plas, K. Pelckmans, B. De Moor, and E. Waelkens. Spatial querying of imaging mass spectrometry data: A nonnegative least squares approach. In *Neural Information Processing Systems Workshop on Machine Learning in Computational Biology*, 2007. pages 82

[109] T. Van Gestel, B. Baesens, J.A.K. Suykens, M. Espinoza, D. E. Baestaens, J. Vanthienen, and B. De Moor. Bankruptcy prediction with least squares support vector machine classifiers. In *Proceedings of the International Conference on Computational Intelligence for Financial Enigeering*, volume 1, pages 1–8, Hong Kong, 2003. pages 9, 35, 36, 66

[110] T. Van Gestel, J.A.K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine Learning*, (1):5–32, 2004. pages 23

[111] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998. pages 2, 17, 18, 60

[112] V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9):2013–2036, 2000. pages 19, 26, 48

[113] G. Wahba. *Spline models for observational data.* CBMS-NSF Regional Conference Series in Applied Mathematics. 59. Philadelphia, PA: SIAM, Society for Industrial and Applied Mathematics. XII, 169 p. , 1990. pages 60

[114] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In *Advances in Neural Information Processing Systems (NIPS'00)*, 2000. pages 25

[115] D. Wettschereck, D.W. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, Feb 1997. pages 24

[116] C. Williams and M. Seeger. The effect of the input density distribution on kernel-based classifiers. In Pat Langley, editor, *ICML*, pages 1159–1166. Morgan Kaufmann, 2000. pages 96

[117] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. volume 13 of *Neural Information Processing Systems (NIPS)*, pages 682–688, 2001. pages 96

[118] M.A. Woodbury. Inverting modified matrices. *Memorandum Rept*, (42):4, 1950. pages 39

[119] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006. pages 82

[120] T. Yun, T. Hwang, K. Cha, and G-S. Yi. CLIC: clustering analysis of large microarray datasets with individual dimension-based clustering. *Nucleic Acids Res*, 38(Web Server issue):W246–53, July 2010. pages 103, 104

[121] R. Zhang and A.I. Rudnicky. A large scale clustering scheme for kernel k-means. In *ICPR (4)*, pages 289–292, 2002. pages 90

[122] X. Zhou and K.Z. Mao. LS bound based gene selection for DNA microarray data. *Bioinformatics*, 21(8), 2005. pages 48, 49, 54, 55

[123] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005. pages 80, 81

List of publications

Journal Papers

- F. Ojeda, J.A.K. Suykens, and B. De Moor. Low rank updated LS-SVM classifiers for fast variable selection. *Neural Networks*, 21(2-3):437–449, 2008.
- J. Luts, F. Ojeda, R. Van de Plas, B. De Moor, S. Van Huffel, and J.A.K. Suykens. A tutorial on support vector machine-based methods for classification problems in chemometrics. *Analytica Chimica Acta*, 665(2):129–145, 2010.
- D. Nitsch, J.P. Gonçalves, F. Ojeda, B. de Moor, and Y. Moreau. Candidate gene prioritization by network analysis of differential expression using machine learning approaches. *BMC Bioinformatics*, 11:460, 2010.
- A. Daemen, O. Gevaert, F. Ojeda, A. Debucquoy, J.A.K. Suykens, C. Sempoux, J.P. Machiels, K. Haustermans, and B. De Moor. A kernel-based integration of genome-wide data for clinical decision support. *Genome Med*, 1(4):39, 2009.

International Conference Papers

- F. Ojeda, M. Signoretto, R. Van de Plas, E. Waelkens, B. De Moor, and J.A.K Suykens. Semi-supervised learning of sparse linear models in mass spectral imaging. In T. Dijkstra, E. Tsivtsivadze, E. Marchiori, and

T. Heskes, editors, *Pattern Recognition in Bioinformatics*, volume 6282 of *Lecture Notes in Computer Science*, pages 325–334. Springer Berlin / Heidelberg, 2010.

- F. Ojeda, T. Falck, B. De Moor, and J.A.K Suykens. Polynomial componentwise LS-SVM: fast variable selection using low rank updates. In *International Joint Conference on Neural Networks (IJCNN 2010)*, pages 3291–3297, July 2010.
- F. Ojeda, J.A.K. Suykens, and B. De Moor. Variable selection by rank-one updates for least squares support vector machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'07)*, pages 2283–2288, Orlando, U.S.A., 2007.
- R. Van de Plas, F. Ojeda, M. Dewil, L. Van Den Bosch, B. De Moor, and E. Waelkens. Prospective exploration of biochemical tissue composition via imaging mass spectrometry guided by principal component analysis. In *Proceedings of the Pacific Symposium on Biocomputing 12*, pages 458–469, Maui, 2007.

Book chapters

- N. Pochet, F. Ojeda[1] , F. De Smet, T. De Bie, J.A.K Suykens, and B. De Moor. Kernel clustering for knowledge discovery in clinical microarray data analysis, chapter III, pages 64–92. In *Kernel Methods in Bioengineering, Signal and Image Processing*. Idea Group Inc., Hershey, Pennsylvania, 2007.

Internal reports

- F. Ojeda, C. Alzate, B. De Moor, J.A.K Suykens. Entropy based selection for spectral clustering. Internal Report 11-83, ESAT-SISTA K.U. Leuven, Leuven, Belgium, 2011.

[1] Also first author.

- A. Daemen, O. Gevaert, F. Ojeda, J.A.K Suykens, and B. De Moor. HI-DID-IT: a HIgh-DImensional Data Integration Toolbox for clinical applications. Internal Report 10-94, ESAT-SISTA K.U. Leuven, Leuven, Belgium, 2010.
- K. De Brabanter, P. Karsmakers, F. Ojeda, C. Alzate, J. De Brabanter, K. Pelckmans, B. De Moor, J. Vandewalle, and J.A.K Suykens. *LS-SVMlab Toolbox User's Guide version 1.7*. ESAT-SISTA K.U. Leuven, Leuven, Belgium, Internal Report 10-146 edition, 2010.
- R. Van de Plas, F. Ojeda, M. Dewil, L. Van Den Bosch, B. De Moor, and E. Waelkens. Unsupervised spatial tissue exploration via imaging mass spectrometry: Preprocessing and clustering. Internal Report 06-32, ESAT-SISTA K.U. Leuven, Leuven, Belgium, 2006.

Abstracts on international conferences/workshops

- F. Ojeda, M. Signoretto, B. De Moor, J.A.K. Suykens. Learning gene networks with sparse inducing estimators. In *MLSB09, Third International Workshop on Machine Learning in Systems Biology*, Jožef Stefan Institute, Ljubljana, Slovenia. September 5-6, 2009.
- F. Ojeda, C. Alzate, J.A.K. Suykens, B. De Moor. A large scale spectral clustering with out of sample extension: application to gene expression data.[2] In *15th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB) & 6th European Conference on Computational Biology (ECCB)*. July 21-25, Vienna, Austria.

[2] *Outstanding poster award.*

Curriculum

Fabian Ojeda was born in Anserma Colombia, on October 1st, 1980. In September 2003, he received his Master's degree in Electronic Engineering from the Universidad Nacional de Colombia, and the Master's degree in Artificial Intelligence (Engineering and Computer Science) from the Katholieke Universiteit Leuven in July 2005. From October 2006, he started his PhD at the Electrical Engineering Department (ESAT/SCD/SISTA) under supervision of Prof. Bart De Moor and Prof. Johan Suykens. His research mainly focuses on applied machine learning, statistical inference and optimization methodologies.

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Electrical Engineering
ESAT-SCD-SISTA Signals, Identification, System Theory and Automation
Kasteelpark Arenberg 10 , B-3001 Heverlee