From the Institute of Neuro- and Bioinformatics
of the University of Lübeck
Director: Prof. Dr. rer. nat. Thomas Martinetz

The Support Feature Machine: An Odyssey in High-dimensional Spaces

Dissertation
for Fulfilment of Requirements for the Doctoral Degree
of the University of Lübeck
from the Department of Computer Sciences/Engineering

Submitted by
Sascha Klement
from Kiel

Lübeck 2013

First referee	Prof. Dr. Thomas Martinetz
Second referee	Prof. Dr. Karsten Keller
Third referee	Prof. Dr. Michael Biehl
Date of oral examination	June 19, 2013
Approved for printing	Lübeck, October 31, 2013

Contents

-Pooh! Buck Mulligan said. We have grown out of Wilde and paradoxes. It's quite simple. He proves by algebra that Hamlet's grandson is Shakespeare's grandfather and that he himself is the ghost of his own father.

«Ulysses», James Joyce

Abstract

Today, researchers and practitioners in diverse fields such as cancer classification, genome analysis, or neuroscience are equipped with highly sophisticated data acquisition devices that produce hard to analyse high-dimensional data. Due to practical or financial issues the number of samples acquired by such systems remains comparatively low — seldom more than a few hundred. Thus, dedicated methods for analysing high-dimensional small sample size data are required. We analyse when and why standard machine learning methods such as the support vector machine may fail to produce proper results on these datasets and motivate why reducing the number of input features to a minimum is absolutely necessary. Therefore, we propose the support feature machine (SFM) as an effective and efficient classifier with inherent feature selection capabilities. The SFM relies on approximation of the zero-norm minimising weight vector of a separating hyperplane by minimising the weight vector's one-norm. A lower number of features is obtained compared to support vector-based feature selection which can be shown both theoretically and empirically. First, we evaluate the SFM's capability to deal with high-dimensional small sample size data and compare it to other methods using artificial data and a genetic benchmark dataset. Then, we show that, with some extensions, the SFM is able to decode brain states in a motor task and even emotional brain states from human functional magnetic resonance imaging (fMRI) data across multiple participants. Further, with the SFM it was possible to quantify the total number of voxels that are informative for discriminating brain states. We found that affective states are represented in whole brain regions — in contrast to classical neurological findings that propose local emotional regions. Additionally, affective states spread over time, i.e. the redundancy of emotional information increases the longer we express an emotion. In summary, we qualify the SFM as a universal method for feature selection — especially promising for advanced analysis of fMRI data.

- Ach was, sagte Buck Mulligan. Oscar Wilde und die Paradoxa haben wir hinter uns. Die Sache ist ganz einfach. Er weist per Algebra nach, dass Hamlets Enkel Shakespeares Großvater ist und er selber der Geist seines eigenen Vaters.

«Ulysses», James Joyce

Zusammenfassung

In der Krebsforschung, Genomanalyse oder in den Neurowissenschaften stehen Wissenschaftlern und Anwendern komplexe Messgeräte zur Datenaufnahme zur Verfügung — die Daten sind stets hochdimensional und erfordern eine aufwendige Datenverarbeitung. Organisatorische, technische und finanzielle Rahmenbedingungen begrenzen die Anzahl der gemessenen Proben auf einige wenige und erfordern spezielle Methoden, um derartige hochdimensionale Daten von geringem Stichprobenumfang zu analysieren. Wir zeigen, wann und warum maschinelle Lernverfahren, wie die Support Vector Machine, nicht in der Lage sind, valide Vorhersagen auf Basis derartiger Daten zu machen [Klement et al., 2008]. Folglich sollte die Anzahl der Merkmale eines Datensatzes stets auf ein Minimum reduziert werden. Dazu haben wir die Support Feature Machine (sfm) entwickelt, eine effektive und effiziente Methode zur Merkmalsselektion. Die sfm basiert auf der Approximation der Null-Norm des Normalenvektors der trennenden Hyperebene durch Minimierung der Eins-Norm. Die Überlegenheit dieses Verfahrens gegenüber Support Vector Verfahren lässt sich sowohl theoretisch wie auch empirisch zeigen [Klement and Martinetz, 2010b, Klement and Martinetz, 2010a, Klement and Martinetz, 2011]. Mit wenigen Erweiterungen ist die sfm in der Lage, Bewegungen und sogar emotionale Zustände probandenübergreifend allein auf der Basis von funktioneller Magnetresonanztomografie (fmrt) vorherzusagen [Klement et al., 2013]. Weiterhin ist es mit der sfm möglich, die Gesamtzahl von Voxeln zu bestimmen, die Information zur Unterscheidung von Hirnzuständen tragen. Damit lässt sich zeigen, dass emotionale Zustände in Mustern kodiert sind, die über das gesamte Gehirn verteilt sind — entgegen der klassischen Sicht von lokalen Emotionsregionen. Außerdem ist die Redundanz emotionaler Information zeitabhängig: Je länger wir uns in einem emotionalen Zustand befinden, desto redundanter wird die Information im Gehirn kodiert. Mit der sfm haben wir eine universelle Methode zur Merkmalsselektion entwickelt, die insbesondere zur Analyse von fmrt Daten geeignet erscheint.

Publikationen

[Klement and Martinetz, 2010a] Klement S and Martinetz T. *A new approach to classification with the least number of features*. In *9th International Conference on Machine Learning and Applications*, pages 141–146. IEEE Computer Society, 2010a.

[Klement and Martinetz, 2010b] Klement S and Martinetz T. *The support feature machine for classifying with the least number of features*. In KI Diamantaras, W Duch, and LS Iliadis, editors, *ICANN (2)*, volume 6353 of *Lecture Notes in Computer Science*, pages 88–93. Springer, 2010b.

[Klement and Martinetz, 2011] Klement S and Martinetz T. *On the problem of finding the least number of features by L1-norm minimisation*. In T Honkela, editor, *Proceedings of the 21st International Conference on Artificial Neural Networks*, Lecture Notes in Computer Science 6791, pages 315–322. Springer, Heidelberg, 2011.

[Klement et al., 2008] Klement S, Madany Mamlouk A, and Martinetz T. *Reliability of cross-validation for SVMs in high-dimensional, low sample size scenarios*. In *Proceedings of the 18th International Conference on Artificial Neural Networks*, pages 41–50. Springer-Verlag, Berlin, Heidelberg, 2008.

[Klement et al., 2011] Klement S, Timm F, and Barth E. *Illumination correction for image stitching*. In *Proceedings of the International Conference on Imaging Theory and Applications*, volume 1, pages 81–86. INSTICC, 2011.

[Klement et al., 2013] Klement S, Anders S, and Martinetz T. *The support feature machine: Classification with the least number of features and its application to neuroimaging data*, 2013. Accepted.

I am a man of constant sorrow
I've seen trouble all my day.
I bid farewell to old Kentucky
The place where I was born and raised.

For six long years I've been in trouble
No pleasures here on earth I found
For in this world I'm bound to ramble
I have no friends to help me now.

From the movie
«O Brother, Where Art Thou?»
directed by Joel and Ethan Coen

Acknowledgements

Ulysses — or Οδυσσέας, as his name is written in Greek — is probably the most famous hero in the ancient Greek myths. His adventurous journey back home after the Trojan War is reported in Homer's epic poem the Odyssey. Themes and characters of the Odyssey are found through the ages in dramas, poems, songs, and even in modern pop culture.

In a wider sense, an odyssey refers to *a long wandering or a voyage usually marked by many changes of fortune* or *an intellectual or spiritual wandering or quest* (Merriam-Webster Online Dictionary. Retrieved March 18, 2011). This perfectly describes — in a metaphorical sense — scientific progress in general and the progress in writing a PhD thesis in special. The Odyssey at hand took almost six years and was indeed marked by many changes of fortune, however, all the wonderful people that surrounded me at that time took away the negative sound that the word Odyssey suggests. I thank all those people who supported me in the last six years and who shaped my way in taking the strange habits in science not too serious: TM — my supervisor; Silke — for providing the fMRI datasets; Erhardt, Fabian, Michael, Martin and Ingrid; Thomas and Mike — the guys from PRC; Dirk, Florian and Foti — the guys from gestigon; Annette — involved in too many businesses to mention; yet most of all my son Jonas and my wife Johanna — for loving me the way I am.

Tell me, O Muse, of that ingenious hero who travelled far and wide after he had sacked the famous town of Troy. Many cities did he visit, and many were the nations with whose manners and customs he was acquainted; moreover he suffered much by sea while trying to save his own life and bring his men safely home; …

«Odyssey », Homer
translated by Samuel Butler

1 Introduction

How do we find the minimal set of features that best describes a certain behaviour when there are countless distracting irrelevant features? This is one of the main questions arising in artificial intelligence, machine learning, neural networks, support vector machines, and statistics. Such learning from examples with many degrees of freedom but few examples is a challenging, yet the most frequent scenario in real-world problems. Today, massively parallel data acquisition systems are standard tools in biological and medical research. They are common in diverse tasks such as tissue classification based on microarray gene data [Golub et al., 1999, Lockhart and Winzeler, 2000], identification of disease-specific genome mutations [Samani et al., 2007, McPherson et al., 2007, Raelson et al., 2007], or information based neuroimaging [Haynes, 2011]. All of them have in common, that practical or financial issues restrict the number of samples to very few.

Some aspects of such *high-dimensional small sample size scenarios* are obvious. First, they can neither be analysed manually nor be visualised in a well-arranged way. Second, the low number of samples can certainly not capture the whole variability of the data. And third, practical issues of automatic computer-based methods — enormous runtime and memory requirements — set limits. Besides, other less obvious aspects make such scenarios hard to handle. The geometry of high-dimensional small sample size data is unintuitive and may cause machine learning methods to produce strange artefacts or to completely fail.

Due to their excellent generalisation capabilities, *maximum margin methods* such as the *support vector machine* (svm) [Vapnik, 1999] have shown to be a good choice for many classification problems in biological and clinical applications. However, these methods may fail especially in high-dimensional small sample size scenarios. Massively parallel data acquisition systems — such as microarrays or mr tomographs — provide many more signals than necessary

to solve a particular task, e.g. deciding whether a specific sample is pathological. Moreover, in biological and clinical applications the primary goal is often not to achieve high prediction accuracy but to identify informative features. Thus, feature selection is not only needed to improve runtime and to achieve proper prediction results, but also to allow meaningful inferences about biologically significant features.

The contribution of this thesis is three-fold. First, we provide novel insight in high-dimensional small sample size data. We show when and prove why the support vector machine may fail to provide proper results. Additionally, we introduce theoretical bounds to measure how likely a dataset may be classified correctly using only few features.

Second, we introduce the *support feature machine* (SFM) as a novel method for feature selection that addresses the above issues: It aims to find the smallest subspace (the least number of features) in a dataset such that within this subspace two classes are linearly separable without error. Thus, results on high-dimensional data become interpretable. And, due to its mathematical formulation, it reduces the influence of high-dimensional artefacts to a minimum. Finally, the engineering task, i.e. the implementation of an SFM is simple and straight-forward — it only requires linear programming solvers, which are available in a variety of flavours, both commercially and free. Results on artificial data as well as real-world datasets demonstrate that this method is able to identify relevant features very effectively and is in many cases superior to SVM-based feature selection approaches, particularly in high-dimensional small sample size scenarios.

Third, the SFM may contribute to some fundamental questions in cognitive neuroscience and neuroimaging. Based on fMRI data it allows to distinguish human brain states, and, further, to quantify the amount and distribution of discriminative information. Our approach supports a recent hypothesis that claims affective information to be distributed in whole brain regions — in contrast to the classical hypothesis of local emotional regions. Even a time-dependent diffusion effect can be observed. Thus, we come closer to understand how affective information is processed in the human brain, however, a universal mindreading device is far from being feasible.

In total, this thesis addresses theoretical issues of high-dimensional data, it introduces and evaluates a novel feature selection method, and it qualifies this method to analyse human brain states.

Thesis Organisation The thesis is organised in five major parts as follows. First, the theoretical basics, frameworks and algorithms are introduced — statistical learning theory, maximum margin methods, feature selection, and statistical geometry. The unintuitive behaviour of high-dimensional small sample size data is analysed in depth to provide insight in why machine

learning methods may fail and which artefacts they may produce. The second chapter introduces the support feature machine as a novel method for feature selection. It covers the theoretical and technical details on how to engineer a support feature machine in an efficient way. The third chapter consists of numerous experiments to verify and compare the performance of the support feature machine. With artificial data and real-world microarray datasets we demonstrate its superiority and practical advantages with respect to support vector-based approaches. After the exclusively machine learning-oriented chapters, we introduce an image processing method for illumination correction based on Gaussian pyramids that is used as a supplementary method in the analysis of volumetric fMRI data. Finally, the fifth chapter describes how the SFM may contribute in understanding human brain activity — especially affective brain states. The thesis concludes with a critical discussion of the results and the impact of the SFM in machine learning and neuroimaging.

The 9000 series is the most reliable computer ever made. No 9000 computer has ever made a mistake or distorted information. We are all, by any practical definition of the words, foolproof and incapable of error.

From the movie
«2001: A Space Odyssey»
directed by Stanley Kubrick

2 Machine Learning and High-dimensional Spaces

A strong mathematical theory is regarded as the best foundation for making any practical apparatus, machinery, instrument, system or technique as foolproof and incapable of error as possible. Because of this, machine learning as a research field has become so popular and successful in recent years. Machine learning provides a variety of tools for classification, regression, density estimation, feature selection, and model estimation, most of which are based on statistical learning theory and structural risk minimisation. The probably most prominent and most widely used method in machine learning is the *support vector machine* (SVM). Although it has been shown theoretically and empirically that the SVM is well suited for classification in many applications, there are also many practical scenarios where it may fail. Especially in high-dimensional small sample size scenarios, which are common in medical and biological applications, it is affected by the enormous amount of irrelevant noise features that are included in the data. Therefore, feature selection methods have been designed to identify relevant and irrelevant features. These feature selection methods come in a variety of flavours mostly aiming to optimise the prediction capability.

This chapter is organised as follows. First, we introduce the mathematical notations that will be used throughout this thesis and briefly introduce the basics of statistical learning theory, structural risk minimisation and support vector learning. For assessing the accuracy of a learning algorithm, we mention standard validation methods and accuracy measures. In the second part, we give an overview of the unintuitive aspects of high-dimensional small sample size scenarios, their geometry and why support vector machines in connection with cross-validation may fail to produces adequate results. Additionally, we give estimates for a random

dataset being linearly separable in the original or a subspace. In certain circumstances, although the data contains no information, we very likely find a low-dimensional subspace in which the data is linearly separable.

2.1 Notations

Typesetting mathematical notations is a science in itself, and there is no universal consensus on the optimal choice — except for not mixing notations. In this work, we use lowercase boldface letters (e.g. $\boldsymbol{x}$, $\boldsymbol{y}$) for *vectors* and uppercase boldface letters for *matrices* (e.g. $\boldsymbol{A}$). *Sets* are typeset in uppercase calligraphic letters (e.g. $\mathcal{D}$).

We make use of the common notations used in classification and feature selection frameworks, i.e. a dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n}$ consists of *feature vectors, samples, patterns* or *data points* $\boldsymbol{x}_i \in \mathbb{R}^d$ and corresponding *class labels* $y_i \in \{-1, +1\}$. The *dimensionality* of a vector is denoted by d, while n refers to the *cardinality* of the set, i.e. the number of data points. For simplicity, we define $\boldsymbol{z}_i = y_i \boldsymbol{x}_i$ and $\boldsymbol{Z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$. The vectors $\mathbf{0}$ and $\mathbf{1}$ are vectors with all their entries being zero or one, respectively. For reasons of readability, we omit the length of these vectors where possible. The *identity matrix* $\boldsymbol{I}_d$ is a square matrix containing ones on the main diagonal and zeros elsewhere, and the *zero matrix* $\mathbf{0}_{n,d}$ has n rows and d columns all set to zero.

A *classifier* $\mathcal{C}$ defines a mapping from the input space to the space of labels. An *inducer* or *induction algorithm* $\mathcal{I}$ builds a classifier $\mathcal{C}$ from a dataset $\mathcal{D}$. A new, unlabelled sample $\boldsymbol{x}$ is classified by

$$\mathcal{I}(\mathcal{D}, \boldsymbol{x}) := (\mathcal{I}(\mathcal{D}))(\boldsymbol{x}) = \mathcal{C}(\boldsymbol{x}) = y\,.$$

The *Kronecker delta* is used to compare whether two variables i and j are equal or not, i.e.

$$\delta_{ij} = \delta(i, j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Exceptions to the above rules are used, if a specific notation is more convenient or due to historical reasons.

2.2 Basics in Statistical Learning Theory

Machine learning, a major branch in artificial intelligence, deals with methods to construct machines with the ability to learn from examples. The statistical learning theory — mainly promoted by Vapnik [Vapnik, 1999] — is a general framework that describes the requirements of successful learning, expected learning performances and appropriate learning strategies.

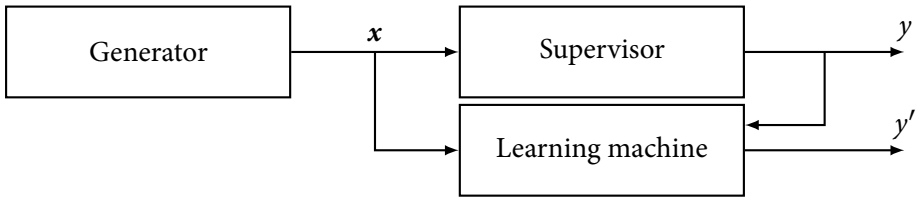


Figure 2.1: Vapnik's model of learning from examples. Generator and supervisor supply the learning machine with the training patterns $\boldsymbol{x}$ and the desired answer y. The learning machine minimises the difference between the supervisor's answer y and the learning machine's answer y'.

Vapnik's function estimation model consists of three components — the *generator*, the *supervisor* and the *learning machine* itself (see Figure 2.1). The generator samples vectors $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ drawn from an unknown but fixed *probability distribution function* $P(\boldsymbol{x})$. The supervisor returns for each input value $\boldsymbol{x}$ an output value $y \in \mathcal{Y}$ according to the unknown *conditional distribution function* $P(y \mid \boldsymbol{x})$. The learning machine implements a set of functions $f(\boldsymbol{x}, \boldsymbol{\alpha}) \in \mathcal{F}$ with parameters $\boldsymbol{\alpha} \in \Lambda$. The joint density $P(\boldsymbol{x}, y)$ is expressed in terms of the *marginal density* $P(\boldsymbol{x})$ and the *conditional density* $P(y \mid \boldsymbol{x})$ by $P(\boldsymbol{x}, y) = P(y \mid \boldsymbol{x}) \cdot P(\boldsymbol{x})$. The *ideal estimator* $f^* \in \mathcal{F}$ minimises the expected error, i.e. the *risk functional*

$$R(\boldsymbol{\alpha}) = \int L(y, f(\boldsymbol{x}, \boldsymbol{\alpha}))\, dP(\boldsymbol{x}, y)$$

if it fulfils

$$f^*(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{\alpha}^*) \quad \text{with} \quad \boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha} \in \Lambda} R(\boldsymbol{\alpha})\,.$$

Here, the *loss function* $L(y, f(\boldsymbol{x}, \boldsymbol{\alpha}))$ describes the difference between the supervisor's and the learning machine's answer. The function space $\mathcal{F}$ is arbitrary, however, it directly controls the generalisation capabilities of the machine and choosing an appropriate function space is a crucial step in machine learning. Depending on the loss function, Vapnik discriminates three machine learning tasks — *classification*, *regression estimation* and *density estimation*. In classification, the task is to discriminate a finite set of classes. The estimated functions $f(\boldsymbol{x}, \boldsymbol{\alpha})$ can only take discrete values — in two-class classification scenarios they are commonly either −1 or +1, however, other values are possible. Then, the loss function

$$L(y, f(\boldsymbol{x}, \boldsymbol{\alpha})) = \begin{cases} 0 & \text{if } y = f(\boldsymbol{x}, \boldsymbol{\alpha}) \\ 1 & \text{if } y \neq f(\boldsymbol{x}, \boldsymbol{\alpha}) \end{cases}$$

indicates whether a pattern was correctly classified by the estimated function or not. In regression estimation, an arbitrary function has to be learned, and the supervisor's answer can take real numbers. A least-squares regression approach uses the loss function $L(y, f(\boldsymbol{x}, \boldsymbol{\alpha})) = (y - f(\boldsymbol{x}, \boldsymbol{\alpha}))^2$, while in density estimation the loss function $L(f(\boldsymbol{x}, \boldsymbol{\alpha})) = -\log f(\boldsymbol{x}, \boldsymbol{\alpha})$ is commonly used. In the following sections the focus will be exclusively on classification.

Empirical Risk Minimisation In practise, the distribution function $P(\boldsymbol{x})$ is not known explicitly but needs to be approximated by a finite set of sample points. Thus, the risk functional $R(\boldsymbol{\alpha})$ is replaced by the *empirical risk functional*

$$R_{\text{emp}}(\boldsymbol{\alpha}) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i, \boldsymbol{\alpha})).$$

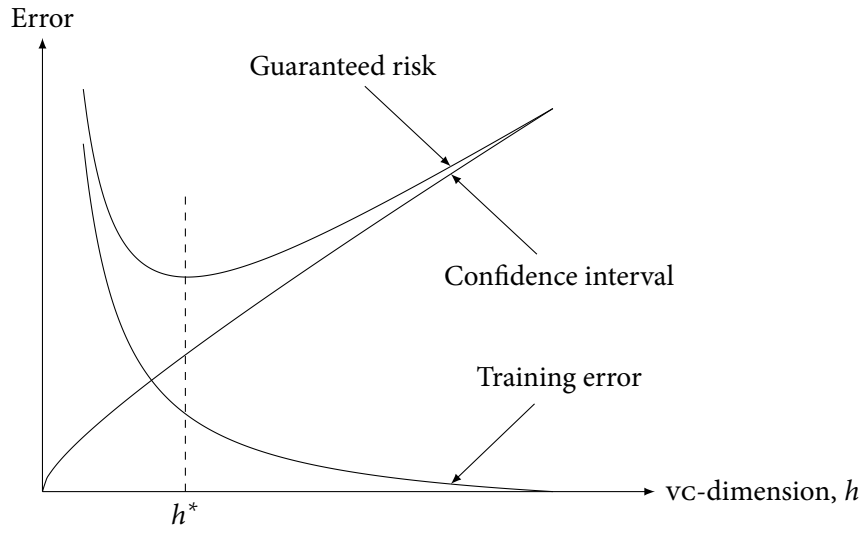
According to the law of large numbers, R_{emp} converges to the *expectation* R with increasing sample size n. However, the arguments that minimise R and R_{emp} are not necessarily the same. In order to find $\boldsymbol{\alpha}^*$ only by minimising R_{emp}, the principle of empirical risk minimisation must be *consistent*, i.e. R and R_{emp} must uniformly converge:

$$\lim_{n \to \infty} P(\sup_{\boldsymbol{\alpha} \in \Lambda} |R(\boldsymbol{\alpha}) - R_{\text{emp}}(\boldsymbol{\alpha})| < \varepsilon) = 0\,.$$

Vapnik-Chervonenkis Dimension Necessary and sufficient conditions for uniform convergence, i.e. consistency, have been derived based on the *Vapnik-Chervonenkis dimension* (VC-dimension) [Vapnik and Chervonenkis, 1982]. This measure describes the expressive power of a family of classification functions. Each dataset $\mathcal{D}$ with n training patterns can be labelled in 2^n different ways, however, not every family of classification functions may correctly separate the two classes for all labellings. Let $N(\mathcal{D}, \mathcal{F})$ be the number of *dichotomies* — i.e. separations into two classes — for the dataset $\mathcal{D}$ that can be realised by a family $\mathcal{F}$ of classification functions. Then, the *growth function*

$$G_{\mathcal{F}}(n) = \max_{\mathcal{D}} N(\mathcal{D}, \mathcal{F}) \leq 2^n$$

is a measure of the maximum number of different labellings for an arbitrary set of size n. The VC-dimension of a function family $\mathcal{F}$ is the maximum number h of patterns, such that these patterns can be separated correctly for each arbitrary labelling. In other words, $\mathcal{C}$ is *shattered* by $\mathcal{F}$. The VC-dimension is infinite if $G_{\mathcal{F}}(n) = 2^n$ for all n. Thus, for any sample size n a particular dataset exists such that the function family can discriminate all different labellings of this dataset. If the VC-dimension is bounded, the growth function is bounded by a polynomial function as



$$\cdots \leq h_{n-1} \leq h_n \leq h_{n+1} \leq \cdots$$

$$\cdots \subset \mathcal{F}_{n-1} \subset \mathcal{F}_n \subset \mathcal{F}_{n+1} \subset \cdots$$

Figure 2.2: Trade-off between training error and confidence interval. The guaranteed risk is an upper bound for the sum of both learning performance measures. Choosing a function class with VC-dimension h^* will yield the smallest guaranteed risk ([Haykin, 1998], slightly modified).

soon as the number of samples exceeds the threshold h (Sauer's lemma, see e.g. [Sauer, 1972]). In this case, no dataset with more than h data points can be shattered. A finite VC dimension is necessary and sufficient for uniform convergence and will guarantee fast convergence [Vapnik and Chervonenkis, 1971, Vapnik and Chervonenkis, 1982]. Thus, learning by minimising the empirical risk will be successful, as the empirical risk converges to the expected risk.

Structural Risk Minimisation According to the VC-theory, the challenge is to define a proper function family that is limited to achieve a low VC dimension but large enough to contain a function that well separates the data. Vapnik proved the generalisation error to be upper bounded by the *guaranteed risk*, which is the sum of the *training error* and the *confidence interval*. The confidence interval is a measure for the probability that a function, taken from the given function family, with small generalisation error can be found at all. The confidence interval increases with increasing VC dimension while the training error decreases (see Figure 2.2). Now, the question is how to determine the function family that yields the least guaranteed risk. The idea of *structural risk minimisation* [Vapnik, 1999] is to define a series of nested *hypothesis spaces*

$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \cdots \subset \mathcal{F}_n$ with increasing VC-dimension, i.e. $h_1 \leq h_2 \leq \cdots \leq h_n$. The learning machine aims to choose the hypothesis space $\mathcal{F}^*$ with the smallest guaranteed risk. In practice, this can be implemented by increasing h until the guaranteed risk does not decrease significantly anymore.

2.3 Support Vector Machines

The family of support vector machines aims to minimise the structural risk by a classifier that maximises the distance — the *margin* — between two classes for a given training dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$. In the most simple case, the classification border is described by a hyperplane defined by a normal vector $\boldsymbol{w}$ and a bias b, i.e. the distance of the hyperplane to the origin. The minimal distance γ from the hyperplane to a pattern is called *geometric margin* (see Figure 2.3). The maximum margin classifier selects that hyperplane among the set of all separating hyperplanes with the largest margin. It can be shown that maximising the margin while enforcing correct classification is equivalent to

$$\begin{array}{ll} \text{minimising} & \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} \\ \text{subject to} & y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geq 1\,, \quad i = 1, \ldots, n\,. \end{array} \tag{2.1}$$

This *primal problem* — a linearly constrained convex optimisation problem — may be solved by quadratic programming. The mathematical formulation has a series of advantages over classical neural networks. Assuming linear separability of the input data, it has a single unique solution — neural networks generally have multiple solutions and may therefore get stuck in local minima during optimisation. Further, the separating hyperplane is exclusively defined by *support vectors*. They are obtained by transforming (2.1) into a *dual formulation* using the *Lagrangian function* that combines objective function and linear constraints and introduces *Lagrangian parameters* α_i for weighting the constraints:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) \quad = \quad \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i \left(y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) - 1\right)\,.$$

Thus, the *dual problem* is to

$$\begin{array}{ll} \text{maximise} & \sum\limits_{i=1}^{n} \alpha_i - \frac{1}{2} \sum\limits_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j \\ \text{subject to} & \begin{cases} \sum\limits_{i=1}^{n} \alpha_i y_i \;=\; 0 \\ \qquad \alpha_i \;\geq\; 0 \quad i = 1, \ldots, n\,. \end{cases} \end{array}$$

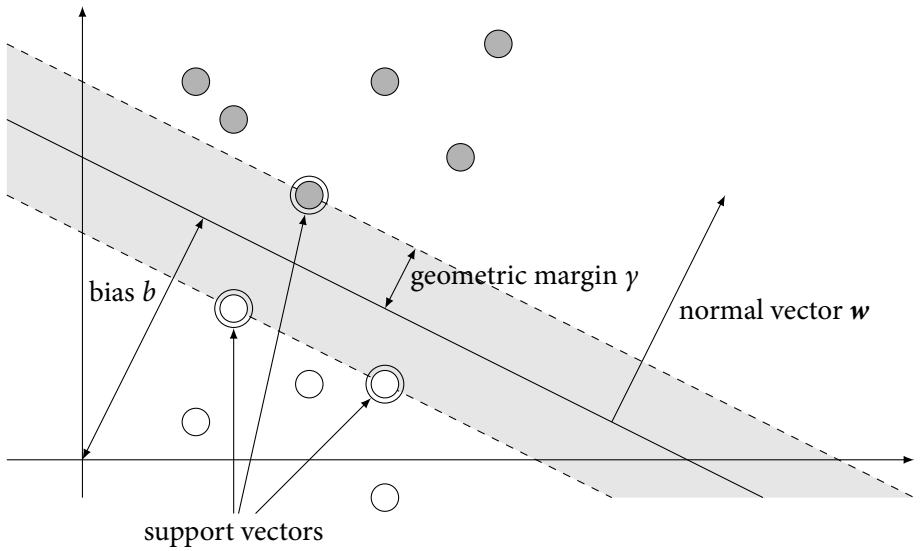


Figure 2.3: Maximum margin classifier. The classifier is defined by a normal vector $\boldsymbol{w}$ and the bias b. Points with geometric margin γ are called support vectors.

The optimal $\boldsymbol{w}^*$ and b^* for the primal problem are obtained from the optimum α^* in the dual representation:

$$\boldsymbol{w}^* = \sum_{i=1}^{n} y_i \alpha_i^* \boldsymbol{x}_i \quad \text{and} \quad b^* = -\frac{1}{2}\left(\max_{i, y_i=-1} \left(\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{w}^* \right) + \min_{i, y_i=+1} \left(\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{w}^* \right) \right).$$

Only those α_i differ from 0 that have a functional margin of +1 or −1. Hence, the corresponding patterns $\boldsymbol{x}_i$ are called support vectors, and all other patterns do not contribute as their Lagrangian parameters are 0:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \left(y_i \alpha_i^* \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x} \right) + b^* = \sum_{i \in S = \{s \mid \alpha_s > 0\}} \left(y_i \alpha_i^* \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x} \right) + b^* .$$

Besides, the dual representation provides a way to introduce the concept of *kernels*, which allow more complicated decision borders to overcome the limitation to linear separable classes. The basic idea of kernels is to transform the low-dimensional input space into a high-dimensional feature space by a mapping $\Phi(\boldsymbol{x})$. As dimensionality increases, a linear hypothesis more likely separates the two classes. In practise, this is achieved by substituting all scalar products $\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$ by a suitable kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^{\mathrm{T}} \Phi(\boldsymbol{x}_j)$. Thus, the transformation $\Phi(\boldsymbol{x})$ into a higher-dimensional space is not done explicitly, but implicitly via the kernel function.

The above *hard-margin classifier* may be strongly affected by outliers — one single outlier may avoid linear separation. Thus, *soft-margin* approaches are favoured in practise. For the *one-norm soft-margin approach*, the dual representation remains the same as in the hard-margin case except for the second constraint, which is now additionally upper bounded by the *softness parameter* C, i.e. $0 \leq \alpha_i \leq C$. In contrast, the *two-norm soft-margin* SVM [Cristianini and Shawe-Taylor, 2000] is implemented using the kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j) = \boldsymbol{x}_i^T \boldsymbol{x}_j + \frac{\delta_{ij}}{C}$$

instead of the dot product in the dual representation. In both approaches, large values of C provide a hard-margin solution, while decreasing the softness parameter allows misclassifications.

As mentioned before, the primal and the dual problem both can directly be solved by quadratic optimisation. However, by taking advantage of the particular structure of the optimisation problem, dedicated methods have been developed, such as *sequential minimal optimisation* [Platt, 1999] or variants of the *MinOver* algorithm [Krauth and Mézard, 1987, Martinetz et al., 2005], which are extensions to the *perceptron* [Rosenblatt, 1958], one of the first artificial neural networks. The *SoftDoubleMinOver* algorithm (see Figure 2.4) implements a two-norm soft-margin SVM by iteratively increasing the weights of those patterns with minimal residual

Input : Feature vectors $\boldsymbol{x}_i$, class labels y_i, number of iterations $t_{\max}$
Output: Weight vector $\boldsymbol{\alpha}$, bias b

$\boldsymbol{\alpha} \leftarrow \mathbf{0}$
for $t \leftarrow 1, \ldots, t_{\max}$ **do**
  for $i \leftarrow 1, \ldots, n$ **do**
    $r_i \leftarrow y_i \sum_{j=1}^{n} y_j \alpha_j \left(K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{\delta_{ij}}{C} \right)$
  end
  $i^+ \leftarrow \arg\min_{i, y_i = +1} r_i$
  $i^- \leftarrow \arg\min_{i, y_i = -1} r_i$
  $\alpha_{i^+} \leftarrow \alpha_{i^+} + 1$
  $\alpha_{i^-} \leftarrow \alpha_{i^-} + 1$
end
Recalculate residuals r_i as above
$b \leftarrow \frac{1}{2} \left(r_{i^-} - r_{i^+} \right)$

Figure 2.4: SoftDoubleMinOver algorithm.

margin from both classes. Either, these are the strongest misclassified patterns, or, if all patterns are correctly classified, the patterns that are closest to the decision border. The solution obtained by SoftDoubleMinOver converges with $O(1/t)$ to the exact solution [Martinetz, 2004, Martinetz et al., 2005].

Finally, we address the issue of unbalanced datasets, where one class is represented by significantly more samples than the other class. A standard soft-margin SVM would be biased towards the smaller class — independent of the actual implementation — as the SVM implicitly assumes equal misclassification costs for each data point. In the limit for very soft scenarios, the SVM behaves like a majority classifier and assigns all samples to the larger class. Several concepts have been proposed to deal with this artefact, e.g. undersampling the majority class, synthetic sample generation for oversampling the minority class [Chawla et al., 2002, Akbani et al., 2004], one-class classifiers [Raskutti and Kowalczyk, 2004], or class-specific softness parameters [Veropoulos et al., 1999]. In the latter approach, each data point is associated with a softness value C^+ or C^- depending on the class label. Equal overall misclassification costs for each class are ensured if $C^+ n^+ = C^- n^-$ holds for classes of size n^+ and n^-, respectively.

2.4 Validation Methods

When it comes to comparing the quality of a classifier, we basically need two things. First, a loss function to compare the predicted and the true outcome of the learning algorithm. And second, a validation scheme, i.e. a method to derive the accuracy not only for a single sample but for a whole dataset. In classification tasks, the loss is commonly defined to be 0 if the classifier predicts the correct class and 1 otherwise. The three widely used validation schemes are *holdout estimate*, *cross-validation* and *bootstrapping*.

The *holdout* method partitions the input data into a training set $\mathcal{D}_t$ and a holdout or test set $\mathcal{D}_h$ of size h. The inducer $\mathcal{I}$ is trained on the training set and its accuracy is determined by classifying all samples of the holdout set, i.e.

$$\mathrm{acc_h} = \frac{1}{h} \cdot \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_h} L\left(\mathcal{I}\left(\mathcal{D}_t, \boldsymbol{x}_i\right), y_i\right)$$

with L as the loss function. A large proportion of the data is never used for training, so the inducer cannot gain any information although the data is present. So, the holdout estimate is often too pessimistic. *Random subsampling*, i.e. splitting the input data several times and averaging the accuracies, takes more data into account.

In *k-fold cross-validation*, the input data is randomly partitioned into k equally sized subsets (folds) $\mathcal{D}_1, \ldots, \mathcal{D}_k$. In each training run, all subsets except for one are used for training, while

the accuracy is estimated on the left-out subset:

$$\mathrm{acc}_{\mathrm{cv}} = \frac{1}{n}\sum_{t=1}^{k}\left(\sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}_t} L(\mathcal{I}(\mathcal{D}\backslash\mathcal{D}_t,\boldsymbol{x}_i),y_i)\right).$$

The extreme case where $k = n$ is called *leave-one-out cross-validation*. If the subsets $\mathcal{D}_i$ are sampled in a naïve way, the class ratios may differ significantly for each fold — especially if the sample size is low — and might bias the accuracy estimate. Such balancing artefacts are avoided by using *stratified cross-validation*, i.e. all folds are sampled to contain the same proportion of class labels.

In *bootstrapping*, the training set is selected by randomly sampling n instances from the input data of size n *with replacement*. Thus, the probability of a sample not to be chosen is $(1-\frac{1}{n})^n \approx e^{-1} \approx 0.368$. The accuracy for a bootstrap sample $\mathcal{D}_t$ is estimated by a weighted sum of training accuracy and test accuracy:

$$\begin{aligned}
\mathrm{acc}_{\mathrm{boot}} &= 0.632\cdot\mathrm{acc}_{\mathrm{test}} + 0.368\cdot\mathrm{acc}_{\mathrm{train}} \quad \text{with}\\
\mathrm{acc}_{\mathrm{test}} &= \sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}_t} L(\mathcal{I}(\mathcal{D}\backslash\mathcal{D}_t,\boldsymbol{x}_i),y_i) \quad \text{and}\\
\mathrm{acc}_{\mathrm{train}} &= \sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}_t} L(\mathcal{I}(\mathcal{D}_t,\boldsymbol{x}_i),y_i)\,.
\end{aligned}$$

Commonly, this measure is averaged over several runs. Bootstrapping was originally introduced in [Efron, 1979]; an overview on various bootstrapping variants can be found in [Efron and Tibshirani, 1993].

In practise, we seek for an accuracy estimator with low bias and low variance. However, each estimator may fail in certain scenarios, e.g. when a simple majority voting rule is used for classification [Kohavi, 1995] or if an SVM is used in high-dimensional small sample size scenarios (see Section 2.5.5). Stratified ten-fold cross-validation [Kohavi, 1995] has been found to be well suited for a variety of real-world scenarios and for different induction methods. Bootstrapping seems to have lower variance but a large bias in some scenarios.

Some attempts have been made to give bounds on the accuracy of these estimates, e.g. in [Kearns and Ron, 1997]. According to [Vapnik, 1982], the difference between true and estimated error will be at most $\tilde{O}\left(\sqrt{h/n}\right)$ for a dataset of size n drawn from an arbitrary input distribution and any learning algorithm with VC-dimension h. Note, the $\tilde{O}$-notation — sometimes called soft-O — ignores logarithmic factors as the big-O notation ignores constants, i.e. $\tilde{O}(g(n))$ is shorthand for $O(g(n)(\log g(n))^k)$.

Alternative Performance Measures The *receiver operating characteristic* (ROC) curve is a method in signal detection theory to choose optimal parameters for a classifier. Depending on a single parameter — e.g. the bias of a support vector machine — it relates false positive and false negative rates. The *area under the curve* (AUC) quantifies the overall performance of the classifier. Assume a test set with n^+ and n^- data points from each class, respectively. First, the decision values obtained from the classifier are sorted in ascending order. Let r_i denote the rank of the ith data point from class +1. Then the AUC is estimated as [Hand and Till, 2001]

$$\text{AUC} = \frac{\sum_{i=1}^{n^+} r_i - \frac{n^+(n^-+1)}{2}}{n^+ n^-} .$$

This measure is independent of the decision threshold and the distribution of the class labels [Bradley, 1997]. Formally, it has been shown that using the AUC measure is indeed statistically consistent and better suited for discriminating performance than the classifier's accuracy [Ling et al., 2003].

2.5 Geometry of High-Dimensional Small Sample Size Scenarios

Convergence proofs and asymptotic bounds in statistical learning theory require sufficiently large datasets that properly represent the data distribution. However, in practise this is generally not the case. Real-world datasets are high-dimensional, but only a few samples may be acquired. Such high-dimensional small sample size scenarios are essentially different from their low-dimensional counterparts. As we do not have an intuition of how a two-thousand-and-one-dimensional space looks like, we tend to characterise it in the same way as two or three-dimensional spaces. But these spaces are totally different and their unintuitive aspects distract learning and validation methods in several ways. The fact that machine learning algorithms do not scale well with the number of features is often referred to as the *curse of dimensionality* [Bellman, 1961].

2.5.1 Empty Space Phenomenon

The most obvious aspect of the curse of dimensionality is that the number of data points required to uniformly cover the whole input space increases exponentially with the number of dimensions [Bellman, 1961]. Given a grid with m points in each direction. The 3-dimensional cube has m^3 grid points, a 4-dimensional hypercube has m^4 grid points and so on. A state-of-the-art microarray chip for analysing human genome expression levels contains 54676

probes (Affymetrix, Inc., GeneChip Human Genome U133 Plus 2.0 Array). A hypercube with the same dimensionality as this microarray dataset and with $m = 2$ has $2^{54676} \approx 10^{16549}$ grid points. For comparison, estimates about the size of the observable universe are in the range of 10^{80} atoms (Wikipedia. Retrieved on August 30, 2011 from `http://en.wikipedia.org/wiki/Observable_universe`). So, any dataset in this space can only cover a vanishing small proportion of the whole space.

2.5.2 Distance Concentration

Another well-known effect is that if dimensionality is increased towards infinity, a finite set of points takes a specific deterministic topology. In the limit, the points are located on the vertices of a regular simplex [Hall et al., 2005], i.e. all samples have nearly the same distances to the origin as well as among each other, and they are pairwise orthogonal. This is referred to as *distance concentration*. Additionally, zero-mean samples taken from a Gaussian distribution are commonly not located near the origin. These properties were shown for multivariate standard normal distributions with zero mean and identity covariance matrix but hold under much weaker assumptions as shown in [Ahn et al., 2007]. Here, the authors derive a condition such that a fixed size dataset behaves as if it was drawn from a distribution with identity covariance matrix for $d \to \infty$. This condition is based on the sphericity measure

$$\varepsilon = \frac{\left(\sum_{i=1}^{d} \lambda_i\right)^2}{d \sum_{i=1}^{d} \lambda_i^2}$$

where λ_i denotes the ith eigenvalue of the covariance matrix. If the eigenvalues are sufficiently *diffused*, i.e. if

$$\lim_{d\to\infty} d \cdot \varepsilon = \lim_{d\to\infty} \frac{\left(\sum_{i=1}^{d} \lambda_i\right)^2}{\sum_{i=1}^{d} \lambda_i^2} \to 0$$

then the dataset will show the same unintuitive behaviour as datasets with the identity covariance matrix (see Figure 2.5 for an example using random normal distributed data with identity covariance matrix). Thus, any method that relies on measuring distances between data points may become meaningless. Nearest neighbour based methods have been analysed with respect to such distance concentration with application to high-dimensional databases [Aggarwal et al., 2001a, Beyer et al., 1999]. In such applications, we seek for a given query data point the data point with minimum distance. However, as dimensionality increases the distance to the nearest and to the farthest data point become more and more equal [Beyer et al., 1999] due to distance concentration — even in cases where the dimensions are correlated or the variance of the newly added dimensions converges to zero. Thus, nearest neighbour methods may become

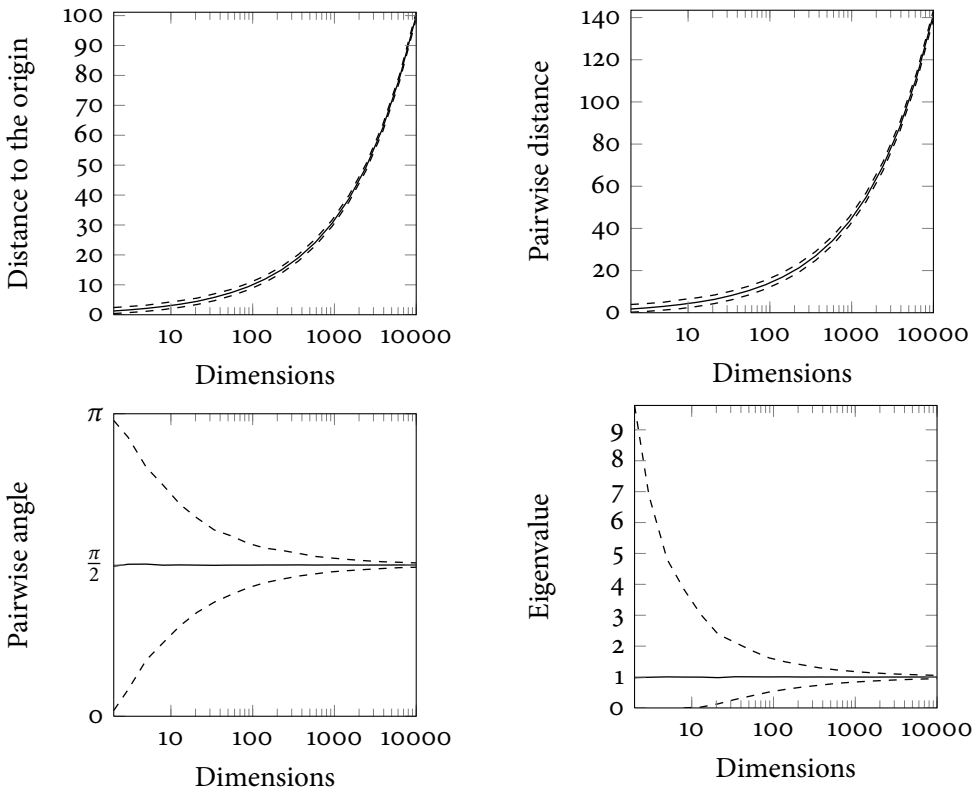


Figure 2.5: Distance concentration in high-dimensional spaces. The effects of distance concentration can be reproduced in a very simple way. Here we sampled $n = 10$ data points normally distributed from low to high-dimensional spaces and plotted the mean (solid) and the extreme values (dashed) for various properties averaged over 100 runs. The distances to the origin (top left) as well as the pairwise distances (top right) concentrate, all pairwise angles (bottom left) converge to 90°, and the eigenvalues of the covariance matrix (bottom right) converge to 1. Thus, distances, angles, and eigenvalues all become the same, although the data was sampled randomly.

meaningless or unstable from 10 to 20 dimensions upwards.

Most nearest neighbour methods apply the *Euclidean norm* as the distance measure, however, other metrics are possible and influence the meaningfulness in high-dimensional spaces [Aggarwal et al., 2001a]. The L_p-norm

$$||\boldsymbol{x}||_p = L_p(\boldsymbol{x}) = \left(\sum_{i=1}^{d} |x_i|^p \right)^{\frac{1}{p}} \quad \text{with} \quad p \in \mathbb{R}, \quad p \geq 1$$

is more susceptible to distance concentration for large values of p. Thus, the best choice with respect to meaningfulness in high-dimensional spaces would be $p = 1$, often referred to as the *Manhattan metric*. Even values between 0 and 1 could be used, however, such *fractional distance measures* are no longer a metric in the strict mathematical sense as the triangle inequality is not fulfilled. However, theoretical and empirical results show, that using fractional distance measures improves the performance of nearest neighbour methods significantly at least on uniformly distributed data [Aggarwal et al., 2001a]. Distance concentration in fractional distance measures may be quantified in terms of *relative concentration*. Let $\boldsymbol{x}$ be a random vector with each feature drawn from some distribution $\mathcal{F}$. Then,

$$\mathrm{RV}_{\mathcal{F},p} = \frac{\sqrt{\mathrm{var}\left(||\boldsymbol{x}||_p\right)}}{\mathrm{E}\left(||\boldsymbol{x}||_p\right)}$$

is a measure of the relative concentration of the norm. Low values indicate a high degree of concentration, high values correspond to a wider distribution of distances. Thus, all distributions and L_p metrics are prone to distance concentration [François et al., 2007] as

$$\lim_{d\to\infty} \frac{\sqrt{\mathrm{var}\left(||\boldsymbol{x}||_p\right)}}{\mathrm{E}\left(||\boldsymbol{x}||_p\right)} = 0.$$

However, the impact depends on the distribution $\mathcal{F}$, and the choice of p needs to be validated for each dataset individually. In total, nearest neighbour methods are prone to the phenomenon of distance concentration, however, there is some evidence that using the L_1-norm for measuring distances relaxes this phenomenon to some extend.

2.5.3 Hubness

Distance concentration is closely related to *hubness* — another high-dimensional artefact that may affect machine learning methods. Hubness refers to the effect that in high-dimensional spaces some data points occur more frequently among the nearest neighbours than others.

Given a dataset $\mathcal{D}$, $N_k(\boldsymbol{x})$ refers to the number of times $\boldsymbol{x}$ is among the k nearest neighbours of all other points in $\mathcal{D}$. In low-dimensional scenarios, N_k converges to a Poisson distribution with mean k, while in the high-dimensional case the distribution of N_k becomes skewed with a long tail to the right [Radovanović et al., 2010]. Thus some data points — *hubs* — occur much more frequent in the list of the k nearest neighbours than others. Hubs have a high tendency to be close to the mean of the data distribution, in multimodal distributions they appear close the mean of the unimodal distribution components. Hubness may occur even after

dimensionality reduction if a distance preserving method is used and the number of features exceeds the intrinsic dimensionality. *Bad hubs*, i.e. hubs with a high probability not having the same class label as the query point, describe the boundary of the classes and thus have a significant impact on classification performance. However, their contribution depends on the induction algorithm. A k-nearest-neighbour classifier can significantly be improved if the contribution of these bad hubs is downweighted as the classifier aims to describe the interior of a class and not its borderline. In contrast, a support vector machine models the separation surface between the classes and, thus, removing bad hubs causes a significant performance drop.

2.5.4 Incidental Separability

In general, a two-class scenario with less samples than features is separable by a linear hyperplane. However, random datasets with more samples than features may be separable by chance. The probability of a dataset being separable by chance depends on the data distribution, the sample size and the dimensionality. In case of rotationally symmetric distributions this probability can be given explicitly — but not for arbitrary distributions. Let $P_{d,n}$ denote the probability of n data points drawn from a d-dimensional distribution to be linearly separable without bias, i.e. the solution needs to pass through the origin. This is equivalent to the probability that all data points are located within the same half-space. For rotationally symmetric distributions, such as the multidimensional standard normal distribution [Wendel, 1962]

$$P_{d,n} = \begin{cases} 2^{-n+1} \sum_{k=0}^{d-1} \binom{n-1}{k} & \text{for } n > d \\ 1 & \text{otherwise}\,. \end{cases}$$

The sample size n needs to be twice as large as the number of features d to let the probability drop to 0.5 (see Figure 2.6). In practise, the above equation allows to estimate whether a linear hard-margin classifier may succeed in finding a solution or not. However, real-world datasets may contain irrelevant noise features and may be separable in less than d dimensions. Again, a purely random dataset may show the same behaviour. Let $P_{d^*,d,n}$ be the probability that a d^*-dimensional subspace with $d^* < d$ exists where all data points are linearly separable or, in other terms, located in the same half-space. As there are $\binom{d}{d^*}$ possible ways to choose the d^*-dimensional subspace, the following upper bound holds [Klement and Martinetz, 2010a]:

$$P_{d^*,d,n} \leq \min\left(1, \binom{d}{d^*} P_{d^*,n}\right) \leq \min\left(1, \binom{d}{d^*} 2^{-n+1} \sum_{k=0}^{d^*-1} \binom{n-1}{k}\right). \tag{2.2}$$

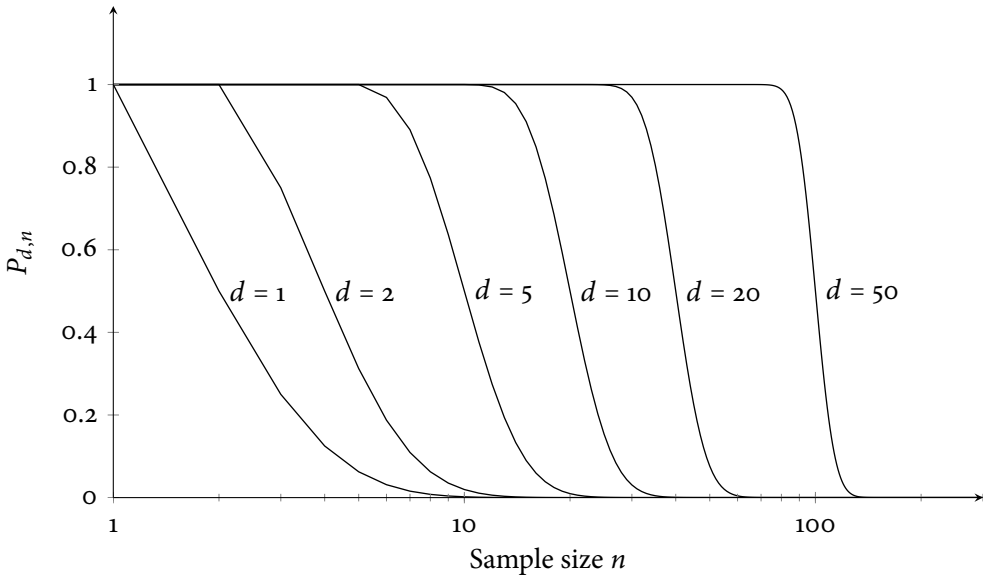


Figure 2.6: Probability of a normal distributed d-dimensional dataset to be located all in the same half-space or being linearly separable without bias.

Additionally, $P_{d^*,d,n} \leq P_{d,n}$ holds, because if the dataset is separable in any subspace, it is also separable in the original space. If it is not separable in the original space, it will never be in any subspace. Further, $P_{d^*,d,n}$ is lower bounded by $P_{d^*,n}$, which can be illustrated as follows: Assume, the dataset to be restricted to d^* dimensions, then obviously $P_{d^*,d^*,n} = P_{d^*,n}$. Adding further dimension may only increase the probability of finding a d^*-dimensional subspace where the data points are separable. In total, the following bounds hold for the probability of a random dataset to be linearly separable in a subspace of dimension d^*:

$$P_{d^*,n} \leq P_{d^*,d,n} \leq \min\left(1, P_{d,n}, \binom{d}{d^*} P_{d^*,n}\right). \tag{2.3}$$

These are very rough estimates and they are constrained to rotationally symmetric distributions. However, if the upper bound is low in an arbitrary scenario, it is very unlikely that a random dataset with the same parameters is separable by chance.

Unfortunately, $P_{d^*,d,n}$ cannot be written in closed form except for the special case $d^* = 1$. Let E_i denote the event that the dataset is separable within dimension i. Now, the probability $P_{1,d,n}$

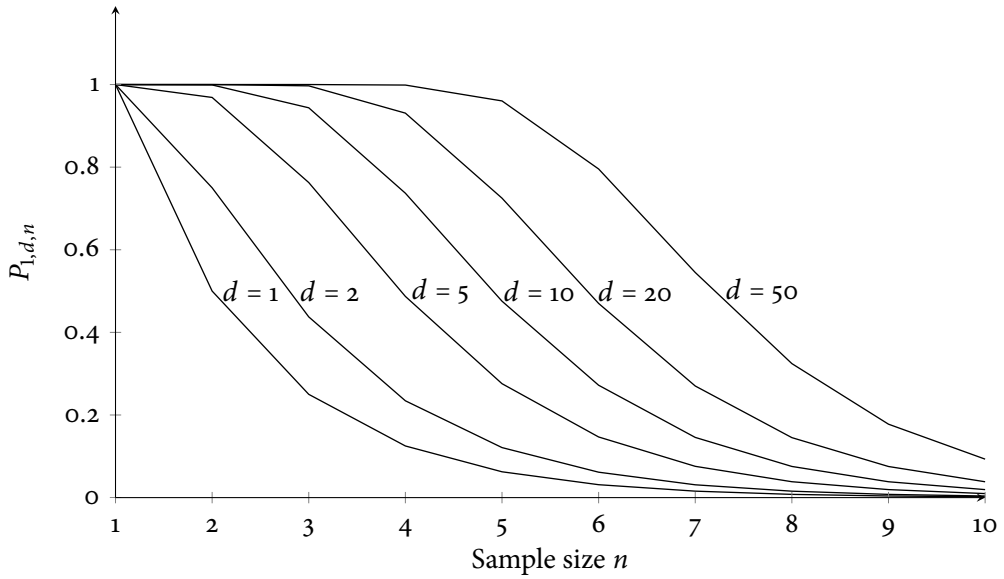


Figure 2.7: Probability of a normal distributed d-dimensional dataset to be linearly separable without bias in any 1-dimensional subspace.

derives to

$$\begin{aligned} P_{1,d,n} &= P\left(\bigcup_{i=1}^{d} E_i\right) \\ &= P(E_1) + \ldots + P(E_d) - P(E_1 \cap E_2) - \ldots - P(E_{d-1} \cap E_d) \\ &\quad + P(E_1 \cap E_2 \cap E_3) + \ldots (-1)^{d-1} P\left(\bigcap_{i=1}^{d} E_i\right) \\ &= \sum_{i=1}^{d} (-1)^{i+1} \binom{d}{i} P_{1,n}^{i} \\ &= \sum_{i=1}^{d} (-1)^{i+1} \binom{d}{i} 2^{i \cdot (-n+1)} . \end{aligned}$$

Here, we use the fact that all events E_i are pairwise statistically independent, i.e. $P(E_i \cap E_j) = P(E_i)P(E_j)$ for all $i \neq j$. The probability $P_{1,d,n}$ drops much faster towards zero (see Figure 2.7) than $P_{d,n}$. Nevertheless, a dataset with 4 samples in 50 dimensions will have at least one dimension in which it is separable with probability 1. Such a ratio of about 1 to 10 may be considered extraordinary large in high-dimensional biological or medical datasets, i.e. high-dimensional real-world data is very likely prone to such behaviour.

Finally, we empirically approximated $P_{d^*,d,n}$ to give an impression of its general behaviour in

various scenarios. Therefore, we sampled d-dimensional datasets with n data points from the standard normal distribution, partitioned them into two balanced classes and tested whether the dataset was separable without bias in any d^*-dimensional subspace. Thus, for each dataset at most $\binom{d}{d^*}$ subsets had to be evaluated. As soon as we found a separable subspace, we skipped the remaining subsets. Besides this combinatorial issue, the question arises which method to choose for testing separability. A non-exhaustive list includes methods based on linear programming, convex hulls, neural networks and quadratic programming [Mangasarian, 1965, Elizondo, 2006]. For sake of simplicity, we chose a method that is as close to the definition of linear separability as possible and does not require any parameters or assumptions. Therefore, we

$$\begin{aligned} &\text{minimise} && \xi \\ &\text{subject to} && y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i \right) + \xi \geq 1 \quad \text{for all } i \\ & && \xi \geq 0 \,. \end{aligned}$$

The dataset is separable if and only if $\xi = 0$ in the optimum [Yogananda et al., 2007]. Of course, we could also train a neural network such as the perceptron and stop the training as soon as separation is achieved. However, the termination criterion, i.e. the number of iterations after which the dataset is classified as inseparable, is hard to choose and highly data dependent.

The empirical results illustrate that in case of low dimensional datasets (e.g., $d = 5$, see Figure 2.8, left column) the bounds are quite close to the empirical probability — for $P_{4,5,n}$ the upper bound almost matches the empirical measurements. In high-dimensional small sample size scenarios, we know that no more than n dimensions are necessary to separate two classes without bias. In general, we do not have an intuition of how likely a separation within very few dimensions may exist. Empirical estimates for medium-sized datasets (see Figure 2.8, right column) are time consuming and become infeasible for arbitrary high-dimensional datasets due to combinatorial issues.

In [Lavine et al., 1988], chance classification has been evaluated empirically depending on the number of data points, the number of features, the class membership distribution and the covariance structure of the data. Based on Monte Carlo simulations, they analysed how likely a certain degree of separability can be achieved on random data. These simulations lead to a simple, yet effective, plausibility check: They suggest to sample multiple instances of random data having the same properties as the original dataset — i.e. the same cardinality, dimensionality, distribution and class balance. The classification results obtained on these random datasets are compared to those of the original dataset, i.e. to the chance level of comparable scenarios. However, this procedure is time consuming as many instances of random data need to be sampled and the classification procedure needs to be executed multiple times to get valid results.

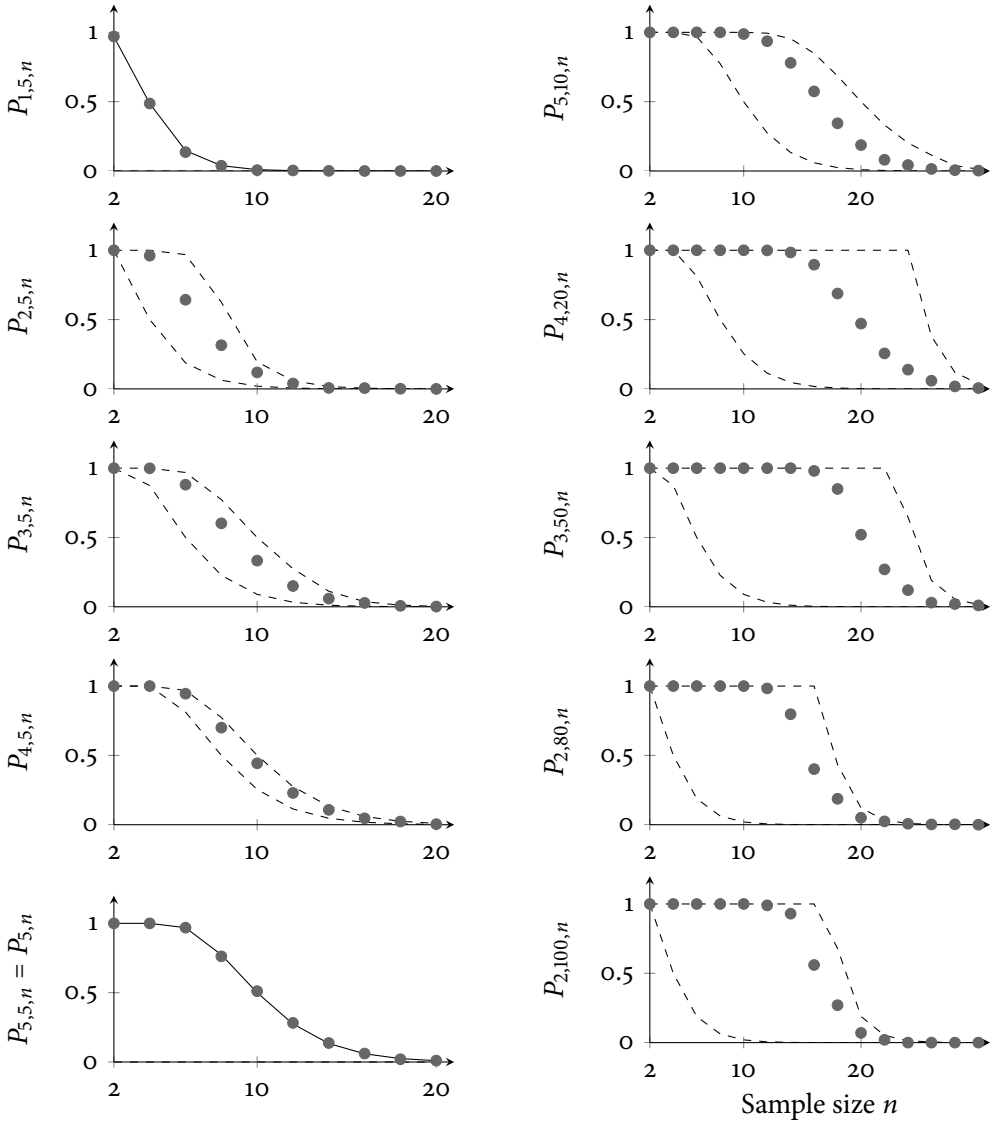


Figure 2.8: Probability of normally distributed d-dimensional data to be linearly separable without bias in any d^*-dimensional subspace. Shown are the empirical probabilities after 1000 repetitions (dots) and the lower and upper bounds (dashed). In the left column, the overall dimension was fixed ($d = 5$) and $P_{d^*,d,n}$ was evaluated for all possible choices of d^*. For $d^* = 1$ and $d^* = 5$ empirical measurements match the bounds, while in all other cases the empirical results are within the bounds. In the right column, various other combinations of d^* and d are shown. However, due to combinatorial issues only those combinations with small $\binom{d}{d^*}$ are included.

2.5.5 Reliability of Cross-Validation

In this section, we analyse when and why cross-validation for support vector machines may be unreliable on high-dimensional small sample size data. Typically, when using the SVM there is a tendency to increase the dimensionality as higher-dimensional datasets are more likely separable. Due to runtime considerations, leave-one-out cross-validation is in general only feasible in small sample size scenarios. So, high dimensionality and small sample size meet if SVMs are validated by leave-one-out cross-validation. The dimensionality of any real-world scenario is finite, however, even comparatively low-dimensional data behaves as if being infinitely dimensional [KLEMENT et al., 2008]. So, *infinity* is rather *small* in small sample size scenarios.

First, we show that the leave-one-out cross-validation error for hard-margin SVMs converges to 1 in high-dimensional feature spaces for equal-sized classes drawn from the same distribution. The expected chance level of 0.5 is only obtained in low dimensions. In the general case — two classes from different distributions — a hard-margin SVM will vote along the majority rule alone for a dimension towards infinity [HALL et al., 2005]. Not only simple hard-margin SVMs are prone to overfitting; soft-margin approaches make things even worse. The margin is increased to reduce the fat-shattering dimension. This is supposed to reduce overfitting by allowing training errors. Unfortunately, this does not increase the generalisation performance, again due to the counterintuitive geometric properties of only few samples in high-dimensional space and the asymmetries of a resampling scheme such as leave-one-out cross-validation. In the soft-margin case, infinity becomes even *smaller*. These properties are proven in the following section.

Random Data Assume a random balanced two-class dataset, i.e. samples drawn from an arbitrary distribution with randomly assigned class labels. The best classifier for completely random datasets is simply the majority voting rule [KOHAVI, 1995]. Unfortunately, leave-one-out cross-validation will indicate poor performance, since the originally balanced dataset becomes unbalanced in each and every validation step. As the left-out pattern reduces the size of one class, a majority classifier will always vote for the other larger class, but the left-out pattern belongs to the smaller class. Thus, the classifier will always make the wrong decision. This behaviour is independent of the dimensionality or training set size.

Such an imbalancing artefact is no particular deficiency of naïve classifiers. In case of high-dimensional scenarios, a linear support vector machine will show the same behaviour. Assume an unlearnable scenario where each feature is independently drawn from the standard normal distribution and the class labels are balanced but randomly assigned. Without loss of generality

we may assign the class labels

$$y_1 = \ldots = y_{\frac{n}{2}} = +1 \quad \text{and} \quad y_{\frac{n}{2}+1} = \ldots = y_n = -1$$

because the support vector machine is independent of the order of the training samples. For high-dimensional small sample size data, $d \gg n$ holds, and therefore, in general, the data is linearly separable, except for cases with three or more collinear data points having alternating class labels. However, the probability of this pathological case is 0. The leave-one-out cross-validation error is determined by training an SVM n times, each time on a different subset of size $n-1$. The obtained classification functions $f_i(\boldsymbol{x})$ are then used to classify the remaining pattern, and the leave-one-out error E is determined by

$$E = \frac{1}{2n} \sum_{i=1}^{n} |f_i(\boldsymbol{x}_i) - y_i| \ .$$

Intuitively, we would expect an average error of 0.5 as the left-out pattern was drawn randomly and is independent of the training data. This is indeed the case in low-dimensional scenarios. However, as dimensionality increases, the error rate converges to 1 for any fixed sample size n (see Figure 2.9). To explain such unintuitive error rates for small sample sizes, we consider datasets with infinite dimensionality.

Theorem 2.5.1 *For any dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ drawn from the multivariate standard normal distribution, with $y_1 = \ldots = y_{\frac{n}{2}} = +1$ and $y_{\frac{n}{2}+1} = \ldots = y_n = -1$ and n fixed, the leave-one-out error rate of a hard-margin* SVM *is* 1 *for $d \to \infty$.*

Proof The proof [Klement et al., 2008] relies on the geometry of high-dimensional datasets as described in Chapter 2.5.2. Namely, for $d \to \infty$ all $\boldsymbol{x}_i \in \mathcal{D}$ lie on the vertices of a regular n-simplex, as well as all pairwise angles are orthogonal. Assuming the data to be drawn from the standard normal distribution with identity covariance matrix, the mean vector length converges to $\sqrt{d}$ as $d \to \infty$ [Hall et al., 2005]. The total variability of $\mathcal{D}$ is provided in the rotation of this simplex. So, without loss of generality, we may rotate the simplex such that the edges are parallel to the coordinate system, i.e we set $\boldsymbol{x}_i = \sqrt{d}\,\boldsymbol{e}_i$. Here, the vectors $\boldsymbol{e}_i$ form the standard basis of the Euclidean space: The ith entry is 1 while all others are 0. Thus, the following properties hold for $d \to \infty$:

$$\begin{aligned} ||\boldsymbol{x}_i||_2 &= \sqrt{d} \qquad \forall\, i \\ ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2 &= ||\boldsymbol{e}_i - \boldsymbol{e}_j||_2 = \sqrt{2\,d} \qquad \forall\, i \neq j \\ \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j &= \boldsymbol{e}_i^{\mathrm{T}} \boldsymbol{e}_j = 0 \qquad \forall\, i \neq j \ . \end{aligned}$$

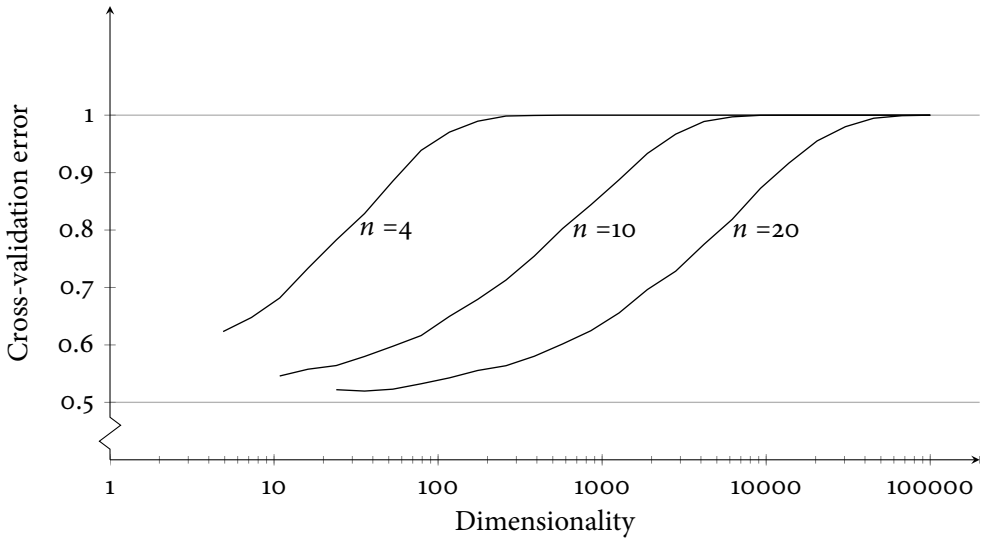


Figure 2.9: Leave-one-out cross-validation error of a hard-margin SVM on normally distributed data with random class labels. Obviously, the error rate converges to 1 for any fixed sample size as the dimensionality goes towards infinity. Only those scenarios were evaluated where $d > n$ to ensure separability. Smooth curves were obtained by averaging 1000 runs.

So indeed, all data points have the same length, the same pairwise distance and they are orthogonal. Again without loss of generality, we select $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}$ for training. We can analytically determine the maximum margin classifier

$$\begin{aligned} f(\boldsymbol{x}) &= \operatorname{sgn}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b\right) \\ \text{that minimises} &\quad \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} \\ \text{subject to} &\quad y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geq 1 \qquad \forall\, i \end{aligned}$$

with simple vector algebra. As all samples are pairwise orthogonal, also the centroids of both classes

$$\boldsymbol{x}^+ = \frac{1}{\frac{n}{2}} \sum_{i=1}^{\frac{n}{2}} \sqrt{d}\, \boldsymbol{e}_i \quad \text{and} \quad \boldsymbol{x}^- = \frac{1}{\frac{n}{2}-1} \sum_{i=\frac{n}{2}+1}^{n-1} \sqrt{d}\, \boldsymbol{e}_i$$

are orthogonal. Thus, the normal vector of the support vector solution points from the centroid of the negative class to the centroid of the positive class (see Figure 2.10), i.e. $\boldsymbol{w} = \alpha\left(\boldsymbol{x}^+ - \boldsymbol{x}^-\right)$. The scaling factor α and the bias are set according to the constraints as follows: Let $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ be arbitrary support vectors from the positive and the negative class, respectively. As they are

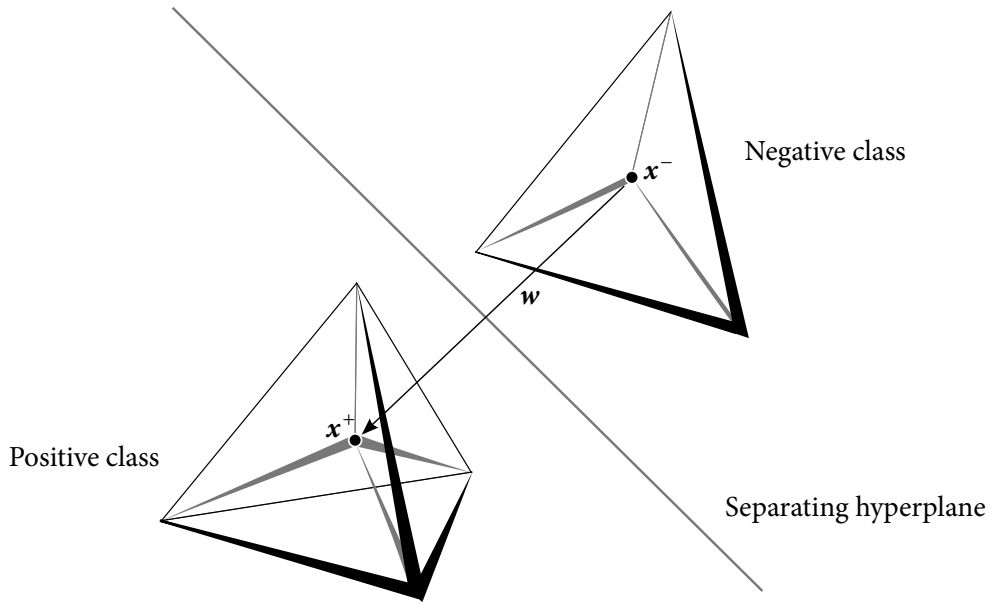


Figure 2.10: Geometry of high-dimensional small sample size data. Here, a schematic view of an infinite dimensional dataset drawn from the multivariate standard normal distribution with 7 data points is shown. The samples from the larger class form a 4-simplex while those from the smaller class form a 3-simplex. The centroids of both simplices are denoted by $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$, respectively. The separating hyperplane with maximum margin has the normal vector $\boldsymbol{w}$ and points from the centroid of the negative class to the centroid of the positive class. However, none of the unintuitive properties of high-dimensional small sample size data can be visualised. For instance, the angle between the segments that join the centre and the vertices of the tetraeder (the 3-dimensional simplex) is $\approx 109.47°$, however for $d \to \infty$ it is 90°.

support vectors, both fulfil the separability constraint with equality:

$$\begin{aligned} y_i \left(\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{w} + b \right) &= 1 \\ y_j \left(\boldsymbol{x}_j^{\mathrm{T}} \boldsymbol{w} + b \right) &= 1 \ . \end{aligned}$$

By substituting the weight vector $\boldsymbol{w}$ and the class labels $y_i = +1$ and $y_j = -1$ we obtain

$$\begin{aligned} \alpha \boldsymbol{x}_i^{\mathrm{T}} \left(\boldsymbol{x}^+ - \boldsymbol{x}^- \right) + b &= \alpha {\boldsymbol{x}_i}^{\mathrm{T}} \boldsymbol{x}^+ - \alpha {\boldsymbol{x}_i}^{\mathrm{T}} \boldsymbol{x}^- + b = \alpha {\boldsymbol{x}_i}^{\mathrm{T}} \boldsymbol{x}^+ + b = 1 \\ -\alpha \boldsymbol{x}_j^{\mathrm{T}} \left(\boldsymbol{x}^+ - \boldsymbol{x}^- \right) - b &= -\alpha {\boldsymbol{x}_i}^{\mathrm{T}} \boldsymbol{x}^+ + \alpha {\boldsymbol{x}_i}^{\mathrm{T}} \boldsymbol{x}^- - b = \alpha {\boldsymbol{x}_j}^{\mathrm{T}} \boldsymbol{x}^- - b = 1 \ . \end{aligned}$$

The solution to this equation system is

$$\alpha = \frac{n \cdot (n-2)}{2\,d \cdot (n-1)} \qquad \text{and} \qquad b = \frac{1}{n-1} \ .$$

Resubstituting α and b shows that all constraints are fulfilled with equality. Therefore, all data points become support vectors, which is anyway an indicator of poor performance. The left-out data point $\boldsymbol{x}_n$ (with $y_n = -1$) is misclassified, because

$$f(\boldsymbol{x}_n) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{e}_n + b = b > 0 \ .$$

This will be the case in each and every validation step for left-out data points x_i with $y_i = -1$. In the opposite case, with one data point left-out from the positive class, the above equations are applied analogously. Again, the left-out data point is misclassified. So the overall leave-one-out cross-validation error is 1. □

With decreasing dimensionality, the data points start to diverge more and more from the vertices of a regular simplex and they are no longer orthogonal. Thus, the conditions of the above proof are only approximately fulfilled. The probability increases for data points to be correctly classified, and the leave-one-out error rate will be less than 1 and will converge to the intuitive error rate of 0.5 for $n \to \infty$.

Real Two-Class Scenarios In the above setting, the data was completely random. But even if both classes are drawn from different distributions, a similar effect may occur. Assume the samples of the two classes to be distributed as $\boldsymbol{\mathcal{X}} = \left(\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(d)}\right)^{\mathrm{T}}$ and $\boldsymbol{\mathcal{Y}} = \left(\mathcal{Y}^{(1)}, \ldots, \mathcal{Y}^{(d)}\right)^{\mathrm{T}}$. The distribution may be arbitrary as long as the mean variances converge for $d \to \infty$:

$$\begin{aligned}
\sigma^2 &= \lim_{d\to\infty} \frac{1}{d} \sum_{i=1}^{d} \operatorname{var}\left(\mathcal{X}^{(i)}\right) \\
\tau^2 &= \lim_{d\to\infty} \frac{1}{d} \sum_{i=1}^{d} \operatorname{var}\left(\mathcal{Y}^{(i)}\right) \\
\mu^2 &= \lim_{d\to\infty} \frac{1}{d} \sum_{i=1}^{d} \left(E\left(\mathcal{X}^{(i)}\right) - E\left(\mathcal{Y}^{(i)}\right)\right)^2 \ .
\end{aligned}$$

Any d-dimensional Gaussian or rectangular distributions fulfil these conditions. Let k and l be the number of data points drawn from $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$, respectively. The following theorem holds for a hard-margin SVM [Hall et al., 2005]:

Theorem 2.5.2 *Assume that $\frac{\sigma^2}{k} \geq \frac{\tau^2}{l}$ (otherwise interchange the classes). If $\mu^2 > \frac{\sigma^2}{k} - \frac{\tau^2}{l}$, then the*

probability of a new datum from either population to be correctly classified by the SVM *converges to* 1 *as* $d \to \infty$. *Otherwise any new sample will be classified as belonging to class* Y *as* $d \to \infty$.

Assume two classes having the same mean variance, i.e. $\sigma^2 = \tau^2$. Further, assume the means of the classes to differ in exactly one dimension and to be the same in all others. Thus, the single relevant dimension has a vanishing impact on the whole difference of the class means, i.e. μ^2 converges to 0 as $d \to \infty$. In leave-one-out cross-validation the class sizes differ by one sample. So, for any fixed sample size n and $d \to \infty$, the condition

$$\frac{\sigma^2}{k} - \frac{\tau^2}{l} = \frac{\sigma^2}{\frac{n}{2} - 1} - \frac{\sigma^2}{\frac{n}{2}} = \frac{\sigma^2}{\left(\frac{n}{2} - 1\right)\frac{n}{2}} > 0 = \mu^2$$

holds. Thus, any new sample is assigned to the larger class, but the left-out sample always belongs to the smaller class. Although the data might be separable in a single dimension, the cross-validation error converges to 1.

Fat-Shattering Dimension The VC-dimension is a measure for the discriminative power of family of classification functions but does not take into account the notion of margin. The generalisation of the VC-dimension to large margin classifiers is the *fat-shattering dimension* $\mathrm{fat}(\gamma)$. It is defined to be the largest number of data points such that all labellings can be discriminated with a margin of γ. The fat-shattering dimension for a linear classifier is upper bounded [Bartlett and Shawe-Taylor, 1999] by

$$\mathrm{fat}(\gamma) \leq \left(\frac{R}{\gamma}\right)^2 \tag{2.4}$$

with R as the radius of the smallest enclosing sphere containing all samples. For random datasets drawn from the standard normal distribution we can derive tight bounds on the fat-shattering dimension [Klement et al., 2008].

Theorem 2.5.3 *For any balanced dataset* $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ *with* $\boldsymbol{x}_i \in \mathbb{R}^d$ *and* $d \to \infty$, *the fat-shattering dimension of a linear support vector machine is bounded by*

$$n - 1 \leq \mathrm{fat}(\gamma) < n \quad \textit{with} \quad \gamma = \sqrt{\frac{d\,(n-1)}{n\,(n-2)}}$$

in each leave-one-out cross-validation step.

Proof Assume the same dataset as in the proof of theorem 2.5.1, i.e. $\boldsymbol{x}_i = \sqrt{d}\,\boldsymbol{e}_i$. For $d \to \infty$

the margin is half the distance between the centroids, i.e.

$$\gamma = \frac{1}{2}\left\|\boldsymbol{x}^{+} - \boldsymbol{x}^{-}\right\| = \sqrt{\frac{d\,(n-1)}{n\,(n-2)}}$$

and the radius of the smallest enclosing sphere containing the complete dataset is $R \leq \sqrt{d}$. Thus, using (2.4) the fat-shattering dimension is upper bounded by

$$\operatorname{fat}(\gamma) \leq \frac{n\,(n-2)}{n-1} < n\,.$$

Further, $\operatorname{fat}(\gamma)$ is lower bounded by $n-1$, if in each validation step all labellings can be shattered with a margin of at least γ. Thus, we need to show, that each labelling of the corners of the $(n-1)$-simplex allows a separation with a margin of γ. Let γ_k be the maximum margin between the classes, if k samples are drawn from the positive class and $n-k-1$ from the negative class:

$$\begin{aligned}\gamma_k &= \frac{1}{2}\left\|\boldsymbol{x}_k^{+} - \boldsymbol{x}_k^{-}\right\| \qquad \text{with} \quad \boldsymbol{x}_k^{+} = \frac{1}{k}\sum_{i=1}^{k}\sqrt{d}\,\boldsymbol{e}_i \quad \text{and} \quad \boldsymbol{x}_k^{-} = \frac{1}{n-k-1}\sum_{i=k+1}^{n-1}\sqrt{d}\,\boldsymbol{e}_i \\ &= \frac{1}{2}\sqrt{\frac{d\,(n-1)}{k\,(n-k-1)}}\,.\end{aligned}$$

The smallest margin γ_k^{*} is obtained for $k^{*} = \arg\max_k k\,(n-k-1)$, by setting the derivative to 0:

$$\frac{\partial(k\,(n-k-1))}{\partial k} = -2\,k + n = 0 \quad \Rightarrow \quad k = \frac{n}{2}\,.$$

Thus, the smallest margin is obtained exactly in the leave-one-out cross-validation setting where the larger class contains $n/2$ samples and the smaller class $n/2-1$ samples. For all other labellings the margin is larger. Therefore, any labelling can be shattered with a margin of at least γ and the fat-shattering dimension is lower bounded by $n-1$. □

These bounds hold for the hard margin case and indicate that no generalisation is possible, as all labellings can be shattered. Extending the above considerations to soft-margin SVMs makes things even worse.

Soft-Margin Hard-margin support vector machines are prone to overfitting because a single outlier may strongly affect the separating hyperplane. Soft-margin SVMs trade off margin maximisation and training error. The two-norm soft-margin SVM is implemented by using the kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^{T}\Phi(\boldsymbol{x}_j) = \boldsymbol{x}_i^{T}\boldsymbol{x}_j + \frac{\delta_{ij}}{C} \tag{2.5}$$

instead of the dot product in the dual representation, where C is the softness parameter. Large values of C cause a hard-margin solution, decreasing the softness parameter allows more and more training errors. Again, assume the same dataset as in the proof of Theorem 2.5.1, i.e. $\boldsymbol{x}_i = \sqrt{d}\,\boldsymbol{e}_i$. The input data is located on the edges of a regular simplex and the kernel function (2.5) preserves this property also in the kernel space [Klement et al., 2008]:

$$\begin{aligned}
\|\Phi(\boldsymbol{x}_i)\| &= \sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i)} = \sqrt{\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i + \frac{1}{C}} = \sqrt{d + \frac{1}{C}} \\
\|\Phi(\boldsymbol{x}_i) - \Phi(\boldsymbol{x}_j)\| &= \sqrt{\big(\Phi(\boldsymbol{x}_i) - \Phi(\boldsymbol{x}_j)\big)^{\mathrm{T}}\big(\Phi(\boldsymbol{x}_i) - \Phi(\boldsymbol{x}_j)\big)} \\
&= \sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i) - 2\,K(\boldsymbol{x}_i, \boldsymbol{x}_j) + K(\boldsymbol{x}_j, \boldsymbol{x}_j)} \\
&= \sqrt{\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i + \boldsymbol{x}_j^{\mathrm{T}}\boldsymbol{x}_j + \frac{2}{C}} \\
&= \sqrt{2\left(d + \frac{1}{C}\right)}.
\end{aligned}$$

All samples are pairwise orthogonal:

$$\Phi(\boldsymbol{x}_i)^{\mathrm{T}}\Phi(\boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j + \frac{\delta_{ij}}{C} = 0 \quad .$$

Thus, a soft-margin SVM on d-dimensional data has the same properties as a hard-margin SVM on $(d + \frac{1}{C})$-dimensional data. For $C \gg 1$ the softness term is negligible, but for $C < 1$ the effective dimensionality increases. The soft-margin approach implicitly increases dimensionality such that the leave-one-out error rate is close to 1 already for a smaller number of dimensions.

At first glance, this is counterintuitive since soft-margin approaches increase the margin and therefore reduce the fat-shattering dimension such that overfitting is reduced. But in case of high-dimensional small sample size data the asymmetries of leave-one-out cross-validation stronger affect generalisation performance than overfitting problems do. For one-norm soft-margin approaches we expect the same behaviour but a closed mathematical formulation cannot be derived as simply as above.

2.6 Feature Selection

The above considerations have shown that high-dimensional small sample size data has many unintuitive geometric characteristics, visualisation is almost impossible, and such data causes a

variety of machine learning algorithms to fail. Even if a method seems to have good prediction performance, it is hard to interprete the results and to say which features are relevant for classification. Therefore, a large family of feature selection methods have been proposed and used in practise, all of them aiming to reduce the number of features while preserving or improving accuracy. The *optimal feature subset* maximises the accuracy of the induced classifier. According to the notion in [Kohavi and John, 1997], each feature may be *strongly relevant*, *weakly relevant* or *irrelevant*, depending on their influence on an optimal Bayesian classifier. A feature is strongly relevant if removing it causes a performance decrease. A weakly relevant feature is one that is not strongly relevant but removing it from a particular feature subset decreases the performance of a Bayesian classifier. All other features are irrelevant.

In the following section, we give an overview on state-of-the-art feature selection methods.

2.6.1 Combinatorial Aspects

The most naïve approach towards feature selection would be *exhaustive search*, i.e. enumerating all $2^d - 1$ feature subsets, training a machine learning method on every single subset, and finally choosing the subset with the optimal performance. Besides the fact that this approach is computationally intractable for most datasets, it has other drawbacks. First, it is not clear which performance measure to use for evaluating whether a feature set is good or not. One might simply use the generalisation performance — approximated by the test error on a separate dataset — aiming for the feature set on which prediction works best. In this case, one could use a validation method such as *k-fold cross-validation* to measure the performance. However, if we aim to find a feature set that allows deeper insight into the data and which allows a better interpretation, we might trade off generalisation performance and the number of features. In this case, the weighting of both measures — test error and feature set size — is the crucial factor.

The second issue arises from the exponential number of training runs with different feature sets on the same initial dataset. Even for random data there is a non-vanishing probability of finding a small subset of features that allows separating the data (see Chapter 2.5.4). So, by enumerating an exponential number of subsets, we might more likely find a good feature set by chance. Thus, all computationally feasible feature selection techniques approximate the optimal feature set, e.g. by Bayesian inference, gradient descent, genetic algorithms, or various numerical optimisation methods.

2.6.2 Categorisation

Feature selection methods may be categorised in several ways — subspace selection vs. dimension reduction, affine vs. combinatorial, and filter vs. wrapper vs. embedded methods.

In *subspace selection*, the task is to find the *largest subset* of features such that the reduced feature set satisfies a property that was not satisfied in the initial set. In contrast, in *dimension reduction* we aim to find the *smallest subset* that still satisfies a property that was also present in the overall dataset [Charikar et al., 2000].

Affine methods allow the selection of any affine subspace — i.e. valid transformations are rotation, scaling, shearing and translation. Such transformations increase the flexibility, however, the results are difficult to interprete with respect to the original feature space. *Combinatorial* feature selection discards some dimensions and preserves the remaining — they are still meaningful with respect to the initial feature space.

The most common taxonomy divides feature selection into *filter*, *wrapper* and *embedded* methods [Saeys et al., 2007]. Filter methods separate feature selection and model selection, the optimal feature subset is selected in advance, i.e. filtered out from the overall set of features without assessing the actual classifier. Wrapper methods evaluate the outcome of the induction algorithm to assess the prediction accuracy on multiple feature sets and apply heuristics to guide the selection process. Embedded methods include feature selection in the model selection such that the obtained classifier is sparse in the sense that only few dimensions are involved. This corresponds to a combined search in the feature and hypothesis space.

2.6.3 Filter Methods

Filter methods use univariate scores to express the discriminative power of a single feature without the need to actually train a classifier. Thus, pure noise features should be eliminated, however, such methods cannot identify multidimensional feature sets with low individual but strong combined discriminative power.

Fisher's discriminant criterion The Fisher Score [Duda et al., 2001] ranks all features according to

$$w_i = \frac{(\mu_i^+ - \mu_i^-)^2}{\left(\sigma_i^+\right)^2 + \left(\sigma_i^-\right)^2}$$

where μ_i^+, μ_i^- and σ_i^+, σ_i^- are the means and the standard deviations for each feature within class +1 and −1, respectively. Large scores indicate good separability.

Golub's Score In [Golub et al., 1999] the authors use

$$w_i = \frac{\mu_i^+ - \mu_i^-}{\sigma_i^+ + \sigma_i^-} \tag{2.6}$$

Input : Feature vectors $\boldsymbol{x}_i$, class labels y_i, number of iterations $t_{\max}$
Output: Weight vector $\boldsymbol{w}$ as a quality criterion for each feature

$\boldsymbol{w} \leftarrow \boldsymbol{0}$
for $t \leftarrow 1, \ldots, t_{\max}$ **do**
  Randomly select a data point $\boldsymbol{x}$
  $\boldsymbol{x}_{\text{hit}} \leftarrow$ nearest neighbour of $\boldsymbol{x}$ having the same class label
  $\boldsymbol{x}_{\text{miss}} \leftarrow$ nearest neighbour of $\boldsymbol{x}$ having opposite class label
  for *all features* j **do**
    $w_j \leftarrow w_j - \frac{1}{t}\operatorname{diff}(j, \boldsymbol{x}, \boldsymbol{x}_{\text{hit}}) + \frac{1}{t}\operatorname{diff}(j, \boldsymbol{x}, \boldsymbol{x}_{\text{miss}})$
  end
end

Figure 2.11: Relief algorithm.

as the ranking criterion. In contrast to the Fisher Score, this criterion can take positive *and* negative values. Positive values indicate correlation with the positive class and vice versa. Besides the ranking criterion, a classification rule can directly be obtained from the weight values:

$$f(\boldsymbol{x}) = \boldsymbol{w}\left(\boldsymbol{x} - \frac{\boldsymbol{\mu}^+ + \boldsymbol{\mu}^-}{2}\right)$$

where $\boldsymbol{\mu}^+$ and $\boldsymbol{\mu}^-$ are the mean vectors within each class.

Relief Ranking The idea of the original Relief algorithm [Kira and Rendell, 1992b, Kira and Rendell, 1992a] is to find those features that can well distinguish instances of the training data that are close together. Within each iteration, the algorithm (see Figure 2.11) randomly selects a data point $\boldsymbol{x}$ and its nearest neighbours from the same class and the opposite class ($\boldsymbol{x}_{\text{hit}}$ and $\boldsymbol{x}_{\text{miss}}$). The quality of a feature — stored in the weight vector $\boldsymbol{w}$ — is increased if $\boldsymbol{x}$ and $\boldsymbol{x}_{\text{miss}}$ largely differ within this feature. If $\boldsymbol{x}$ and $\boldsymbol{x}_{\text{hit}}$ largely differ, the quality is decreased. The update step involves the univariate distance measure

$$\operatorname{diff}(i, \boldsymbol{x}, \boldsymbol{y}) = \frac{|x_i - y_i|}{\max_i - \min_i}$$

where $\max_i$ and $\min_i$ are the minimum and maximum values of feature i in the overall dataset. In the nearest neighbour search, the above measure is applied for every feature and the outcomes are summed to get the overall distance.

2.6.4 Wrapper Methods

In wrapper approaches the set of relevant features is successively updated according to the accuracy of a classifier on subsets of features. It requires the choice of four components — a *state space*, an *initial state*, a *termination condition* and a *search engine* [Kohavi and John, 1997]. The state space consists of all 2^d feature subsets. Two states are connected, if the second state contains the same features as the first state but one additional feature. Other interconnection networks are as well possible. The optimal feature subset is approximated based on heuristics that define the initial state, the termination criterion and the search engine. In *forward selection* the initial state is the empty set, while in *backward elimination* it contains all features. The *hill-climbing* or *greedy* search engine evaluates the accuracy of each child node and then moves to the child node with highest accuracy. This procedure is terminated if the accuracy can no further be improved. In contrast, the *best-first* search engine maintains a list of nodes that have been evaluated so far, ordered according to their accuracy starting with the best. Within each iteration, the neighbours of the first node within this list are evaluated and added to the list. It terminates, if the accuracy has not been improved within a certain number of iterations.

Wrapper methods are an intuitive way to combine feature selection and the entire classification framework, however, fine-tuning the search heuristic may become complex.

2.6.5 Embedded Methods

Recursive Feature Elimination The feature ranking scores obtained by Golub's method can directly be used as weights in a classifier, i.e. the feature selection results induce a linear classifier. The opposite way would be to train a classifier and use the weight vector as a feature ranking criterion. This is exactly what *recursive feature elimination* (rfe) [Guyon et al., 2002] does. It is inspired by the *optimal brain damage* (obd) [LeCun et al., 1990] — a method for successively reducing the number of connections in a neural network by setting those connection weights to zero that cause a minimal performance loss. The loss can be approximated by expanding the cost function locally as a second order Taylor series. For linear svms with quadratic cost function, this corresponds to discarding those features with minimum absolute weight (see Figure 2.12 for the overall algorithm). To reduce runtime, Guyon et al. suggest to discard multiple features within each iteration.

Feature Selection via Mathematical Programming Alternatively, feature selection may be regarded as an optimisation problem where the number of features and the training error are both minimised. Given two classes with the data matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, having m and k samples,

Input : Feature vectors $\boldsymbol{x}_i$ and class labels y_i
Output: List of ranks r_i, weight vector $\boldsymbol{w}$ and bias b

Initialise list of surviving features $\boldsymbol{s} \leftarrow (1 \ \ldots \ d)$
while $\boldsymbol{s} \neq \varnothing$ **do**
  Reduce feature vectors to surviving features, i.e. $\boldsymbol{X}' = \boldsymbol{X}(:, \boldsymbol{s})$
  Train an SVM on $\boldsymbol{X}'$ and $\boldsymbol{y}$, store weight vector $\boldsymbol{w}$
  Compute ranking criterion for all features, i.e. $c_i \leftarrow w_i^2$
  Find feature with minimum rank, i.e. $f \leftarrow \arg\min \boldsymbol{c}$
  Add feature to rank list, i.e. $\boldsymbol{r} \leftarrow \begin{pmatrix} s_f & \boldsymbol{r} \end{pmatrix}$
  Eliminate feature with lowest rank, i.e. $\boldsymbol{s} = \begin{pmatrix} s_1 & \ldots & s_{f-1} & s_{f+1} & \ldots \end{pmatrix}$
end

Figure 2.12: Recursive feature elimination

respectively. The optimisation problem [Bradley and Mangasarian, 1998]

$$\begin{aligned} \text{minimise} \quad & (1-\lambda)\left(\frac{\mathbf{1}^{\mathrm{T}}\boldsymbol{y}}{m} + \frac{\mathbf{1}^{\mathrm{T}}\boldsymbol{z}}{k}\right) + \lambda\, \mathbf{1}^{\mathrm{T}}\boldsymbol{w}_* \quad \text{with} \quad (w_*)_i = \begin{cases} 0 & \text{if } w_i = 0 \\ 1 & \text{otherwise} \end{cases} \\ \text{subject to} \quad & -\boldsymbol{A}^{\mathrm{T}}\boldsymbol{w} + \mathbf{1}\gamma + \mathbf{1} \leq \boldsymbol{y} \\ & \boldsymbol{B}^{\mathrm{T}}\boldsymbol{w} - \mathbf{1}\gamma + \mathbf{1} \leq \boldsymbol{z} \\ & \boldsymbol{y} \geq 0, \quad \boldsymbol{z} \geq 0, \quad -\boldsymbol{v} \leq \boldsymbol{w} \leq \boldsymbol{v} \end{aligned}$$

allows a trade-off between the training error and the number of non-zero entries in the weight vector by choosing an appropriate $\lambda \in [0, 1)$. Here, $\boldsymbol{w}$ denotes the weight vector and γ is the bias such that any new sample $\boldsymbol{x}$ belongs to class A if $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{w} > \gamma$. The vectors $\boldsymbol{y}$ and $\boldsymbol{z}$ are the class-specific slack variables, i.e. they quantify how far each pattern is away from being correctly classified. The objective function may be linearised, and can then be solved by a successive linearisation algorithm. Thus, a classifier with inherent feature selection is obtained.

Feature Scaling Methods for Support Vector Machines Feature scaling based methods iteratively increase the weight of putatively relevant features and decrease the weight of putatively irrelevant features. In case of convergence, this weight vector — not to be confused with the weight vector of the entire classifier — quantifies the relevance of each feature. Most approaches alternate between solving a support vector machine and ranking the features according to some error criterion [Mukherjee et al., 1998, Jebara and Jaakkola, 2000, Chapelle et al., 2002, Weston et al., 2000].

One-Norm Support Vector Machines The one-norm support vector machine [Zhu et al., 2004] is an application of the *lasso* [Tibshirani, 1996] to classification. It

$$\begin{aligned} \text{minimises} \quad & \sum_{i=1}^{n} \left[1 - y_i \left(b + \sum_{j=1}^{q} w_j h_j(\boldsymbol{x}_i)\right)\right]_+ \\ \text{subject to} \quad & \|\boldsymbol{w}\|_1 \leq s \end{aligned}$$

with $[k]_+ = \max(k, 0)$ for a set of basis functions $\{h_i, \ldots, h_q\}$, a weight vector $\boldsymbol{w}$, a bias b and a tuning parameter s. A more SVM-like notation is obtained using the original features instead of basis functions — $h_j(\boldsymbol{x}) = x_j$ — and a dual representation:

$$\text{minimise} \sum_{i=1}^{n} \left[1 - y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right)\right]_+ + \lambda \|\boldsymbol{w}\|_1 .$$

Zero-norm Based Methods All the above support vector related methods enforce sparse solutions by adding parameters or constraints to the standard SVM optimisation procedure. Thus, sparsity is merely an effect than a primary aim. In contrast, one may approximate the zero-norm minimising weight vector of a separating hyperplane directly [Weston et al., 2003]. We assume the dataset $\mathcal{D}$ to be linearly separable, i.e.

$$\exists\, \boldsymbol{w} \in \mathbb{R}^d,\ b \in \mathbb{R} \quad \text{with} \quad y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) \geq 0 \ \ \forall\, i \quad \text{and} \quad \boldsymbol{w} \neq \mathbf{0}\,, \tag{2.7}$$

where the normal vector $\boldsymbol{w} \in \mathbb{R}^d$ and the bias $b \in \mathbb{R}$ describe the separating hyperplane except for a constant factor. Obviously, if $\boldsymbol{w}$ and b are solutions to the inequalities, also $\lambda\, \boldsymbol{w}$ and $\lambda\, b$ solve them with $\lambda \in \mathbb{R}^+$. In general, there is no unique solution to (2.7). A solution with the least number of features

$$\begin{aligned} \text{minimises} \quad & \|\boldsymbol{w}\|_0^0 \\ \text{subject to} \quad & y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) \geq 0 \quad \forall\, i \\ \text{and} \quad & \boldsymbol{w} \neq \mathbf{0} \end{aligned} \tag{2.8}$$

with $\|\boldsymbol{w}\|_0^0 = \text{card}\,\{w_i | w_i \neq 0\}$. Note, that any solution of (2.8) can be multiplied by a positive factor and is still a solution. Weston et al. proposed to solve the above problem with a variant of the support vector machine by

$$\begin{aligned} \text{minimising} \quad & \|\boldsymbol{w}\|_0^0 \\ \text{subject to} \quad & y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) \geq 1 \quad \forall\, i\,. \end{aligned} \tag{2.9}$$

Indeed, as long as there exists a solution to (2.8) for which $y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) > 0$ for all $i = 1, \ldots, n$, solving (2.9) yields a solution to (2.8). Unfortunately, (2.8) as well as (2.9) are NP-hard and

Input : Feature vectors $\boldsymbol{x}_i$ and class labels y_i
Output: Weight vector $\boldsymbol{w}$ and bias b

Initialise $\boldsymbol{z} = (1, \ldots, 1)$.
repeat
  Minimise $|\boldsymbol{w}|$ such that $y_i \left(\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{x}_i * \boldsymbol{z}) + b\right) \geq 1$
  Update $\boldsymbol{z} = \boldsymbol{z} * \boldsymbol{w}$
until *convergence*

Figure 2.13: Iterative zero-norm approximating algorithm according to [Weston et al., 2003].

cannot be solved in polynomial time. Therefore, Weston et al. proposed to approximate (2.9) by

$$\begin{aligned} \text{minimising} \quad & \textstyle\sum_{j=1}^{d} \ln\left(\varepsilon + |w_j|\right) \\ \text{subject to} \quad & y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) \geq 1 \quad \forall\, i \end{aligned} \tag{2.10}$$

with $0 < \varepsilon \ll 1$. If $\boldsymbol{w}_0$ and $\boldsymbol{w}^*$ optimise (2.9) and (2.10), respectively, then

$$\|\boldsymbol{w}^*\|_0^0 \leq \|\boldsymbol{w}_0\|_0^0 + \mathcal{O}\left(\frac{1}{\ln \varepsilon}\right), \tag{2.11}$$

i.e. both solutions coincide as $\varepsilon \to 0$. Thus, by minimising (2.10) an approximate solution to (2.9) is found. However, (2.10) is not convex, may have many local minima, and is still hard to solve. Weston et al. proposed an iterative scheme (see Figure 2.13) which finds a local minimum of (2.10) by solving a sequence of linear programs. This modification of the support vector machine effectively reduces the feature space used for classification. However, the number of features may be further reduced by discarding any margin maximisation induced by the constraints $y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) \geq 1$. This is the basic idea of the support feature machine proposed in the next chapter.

2.7 Conclusions

Statistical learning theory and support vector based methods are well established research fields, but their paradigms may fail in high-dimensional small sample size scenarios. Such datasets are prone to the empty space phenomenon, distance concentration, hubness and incidental separability. Further, support vector classification may produce completely unintuitive leave-one-out cross-validation errors. Therefore, irrelevant features need to be excluded from the training

data, or, if no prior information about relevance is available, feature selection methods should be used for preprocessing. Here, multidimensional embedded methods are most promising as feature selection and classification are directly linked. Finally, there is some evidence that zero-norm based approaches are well suited for feature selection in high-dimensional spaces, as the phenomenon of distance concentration becomes less prominent.

Mr Leopold Bloom ate with relish the inner organs of beasts and fowls. He liked thick giblet soup, nutty gizzards, a stuffed roast heart, liverslices fried with crustcrumbs, fried hencods' roes. Most of all he liked grilled mutton kidneys which gave to his palate a fine tang of faintly scented urine.

«Ulysses», James Joyce

3 Support Feature Machine

The support feature machine (SFM) — the main contribution of this work — is a novel method for feature selection and aims to find the smallest subspace (the least number of features) such that in this subspace two classes are linearly separable without error. Such a minimal feature subset is essentially useful for the interpretation of high-dimensional data, e.g. in biological or clinical applications, where we not only aim to classify samples but rather want to infer novel insight on how the data is structured. In the terminology of feature selection, it is an embedded method that combines feature selection and classification within a single framework. The SFM implements the principle of structural risk minimisation (see Chapter 2.2) by limiting the family of classification functions to those with the fewest number of parameters — in this case dimensions. Minimising the number of features also minimises the chance of incidental separability (see Chapter 2.5.4). Most feature selection methods are conservative in the way they select features, i.e. they keep all features that could ever be relevant and discard only those features that are irrelevant for classification with high probability. In contrast, the SFM is very aggressive in discarding features.

First, we motivate and define the core support feature machine for linearly separable datasets. We then extend the basic SFM to be applicable to non-separable classes and unbalanced datasets and show how a repetitive SFM algorithm might be used to estimate the proportion of informative features in highly redundant datasets. Further, the superiority of the SFM with respect to the closely related SVM-based method by Weston et al. is made plausible in the second part of this chapter. Further, we derive bounds on the VC-dimension of combined feature selection and classification to estimate the expressive power of the SFM. Finally, an SFM implementation requires solving a linear program. Therefore, the basic SFM notation needs to be transformed into a solver-specific notation. We provide two alternative SFM formulations that differ in size and sparsity of the linear program.

3.1 Basic Algorithm

Minimising the number of features by minimising the zero-norm of the weight vector of the separating hyperplane has previously been proposed [Weston et al., 2003], however, their approach performs a mixture of feature selection and margin maximisation, which might be conflicting objectives — a larger margin could induce a larger number of features. Taking a different approach [Klement and Martinetz, 2010b], we adapt the definition of linear separability (2.8) slightly such that we

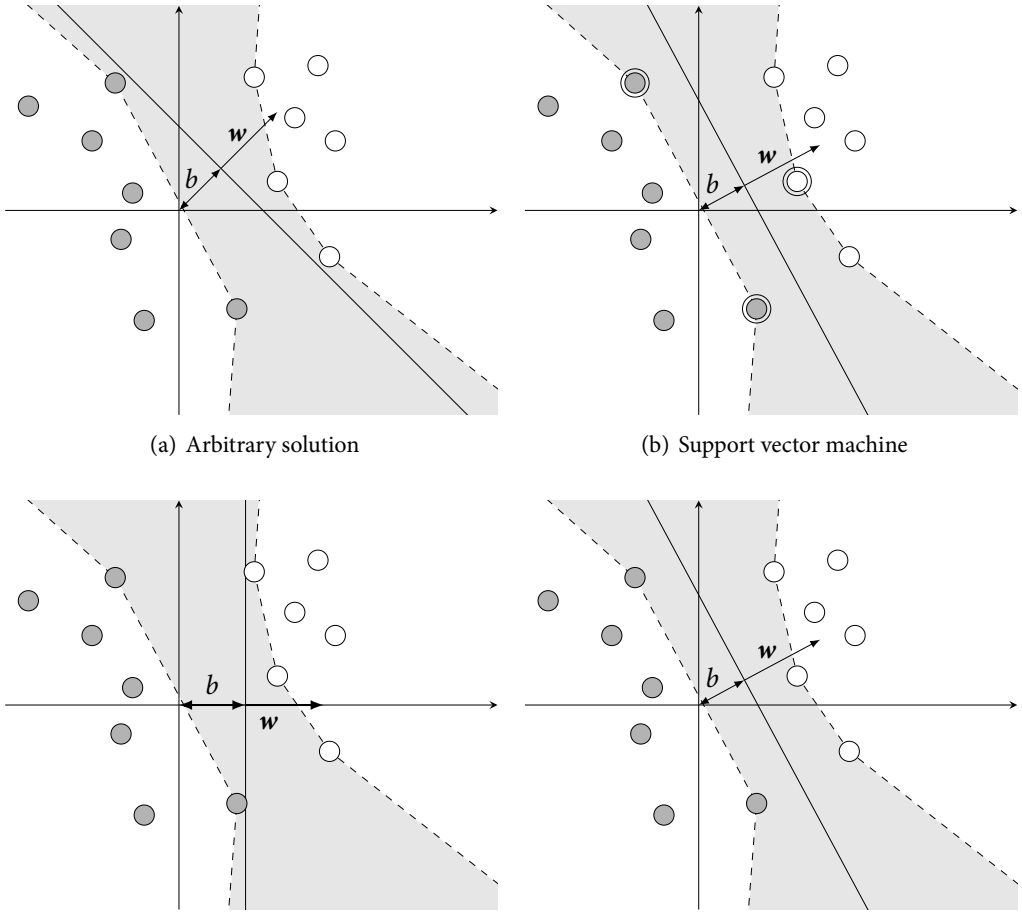
$$\begin{aligned} \text{minimise} \quad & \|\boldsymbol{w}\|_0^0 \\ \text{subject to} \quad & y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) \geq 0 \quad \forall\, i \\ \text{and} \quad & \tfrac{1}{n} \textstyle\sum_{i=1}^{n} y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) = 1 \ . \end{aligned} \tag{3.1}$$

The first constraint is insensitive to any margin. The second constraint excludes the trivial solution $\boldsymbol{w} = \boldsymbol{0}$, since otherwise we would obtain $\frac{1}{n} \sum_{i=1}^{n} y_i b = 1$ and $y_i b \geq 0$, which cannot be fulfilled for all i, because we have labels +1 and −1. As long as the input data is linearly separable with $y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) > 0$ for at least one $i \in \{1, \ldots, n\}$, the second constraint $\frac{1}{n} \sum_{i=1}^{n} y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) = 1$ can be satisfied by scaling $\boldsymbol{w}$ and b appropriately. Hence, solving (3.1) yields a solution to the ultimate problem (2.8). See Figure 3.1 for a 2-dimensional example that illustrates the differences between support vector machine, support feature machine, and the SVM-based feature selection method by Weston et al. In this scenario Weston's method is unable to find the optimal solution that involves a single feature.

Weston et al. apply an iterative framework to find an approximate solution to their formulation of a zero-norm SVM (2.9). They reformulate the objective function $\|\boldsymbol{w}\|_0^0$ as $\sum_{j=1}^{d} \ln\left(\varepsilon + |w_j|\right)$ such that an approximate solution to the minimisation problem can be found by gradient descent. Thus, their zero-norm approximation framework can be applied in the same way to solve (3.1), i.e. we

$$\begin{aligned} \text{minimise} \quad & \textstyle\sum_{j=1}^{d} \ln\left(\varepsilon + |w_j|\right) \\ \text{subject to} \quad & y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) \geq 0 \quad \forall\, i \\ \text{and} \quad & \tfrac{1}{n} \textstyle\sum_{i=1}^{n} y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b\right) = 1 \end{aligned} \tag{3.2}$$

with a similar iterative scheme (see Figure 3.2). By successively minimising the one-norm we aim to approximate the zero-norm minimising solution as accurately as possible. This implements the principle of structural risk minimisation: In the very first iteration the unscaled d-dimensional input data is used for training. This corresponds to finding a solution in an unrestricted hypothesis space $\mathcal{F}_d$ with VC-dimension h_d. Here, $h_d = d + 1$ as we are limited to linear classifiers with bias. If the weight vector contains zero entries after the first iteration, then the corresponding entries of the scaling vector $\boldsymbol{z}$ are set to zero, and these features will have



(a) Arbitrary solution

(b) Support vector machine

(c) Support feature machine

(d) Weston's method

Figure 3.1: Solutions obtained by different classifiers. The given dataset (white dots: class +1, gray dots: class −1) is linearly separable in many ways — any classifier passing exclusively the gray region with the weight vector pointing towards the positive class is a valid solution, e.g. the classifier in (a). The support vector classifier (b) separates both classes with maximum margin and is uniquely defined by the support vectors (marked by circles) that all have the same distance to the decision border. In contrast, the support feature machine (c) minimises the number of involved features — in this example one feature (the x-axis) is sufficient to linearly separate the classes. Note, the solution is not unique; the classifier may be shifted slightly without classification error. For this dataset, the SVM-based feature selection method by Weston et al. (d) produces the same solution as a standard SVM, i.e. it is unable to find the least number of features for separation.

Input : Feature vectors $\boldsymbol{x}_i$ and class labels y_i
Output: Weight vector $\boldsymbol{w}$ and bias b

1 Initialise $\boldsymbol{z} = (1, \ldots, 1)$
2 **repeat**
3 | Minimise $|\boldsymbol{w}|$ such that $y_i \left(\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{x}_i * \boldsymbol{z}) + b\right) \geq 0$ and $\frac{1}{n}\sum_{i=1}^{n} y_i \left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) = 1$
4 | Update $\boldsymbol{z} = \boldsymbol{z} * \boldsymbol{w}$
5 **until** *convergence*

Figure 3.2: Iterative SFM algorithm. The algorithm aims to minimise the zero-norm of the weight vector $\boldsymbol{w}$ by iteratively minimising its one-norm. The operator $*$ denotes the component-wise multiplication.

no further effect on the training. The hypothesis space is reduced to $\mathcal{F}_{d'}$ with $\mathcal{F}_{d'} \subset \mathcal{F}_d$ and a reduced VC-dimension $h_{d'}$. Thus, the SFM derives a set of nested hypothesis spaces, reduces the VC-dimension, and, therefore, minimises the structural risk.

Connection to Sparse Coding The support feature machine is a remote relative of the family of *sparse coding* concepts. The SFM aims to minimise the number of features to distinguish two classes, i.e. it seeks a separating hyperplane with the sparsest weight vector. In sparse coding, the task is to represent a signal by a linear combination of basis functions with as few components as possible, i.e. we seek the sparsest representation of a signal.

Given a signal $\boldsymbol{x} \in \mathbb{R}^d$ and a *dictionary* $\boldsymbol{C} \in \mathbb{R}^{d \times l}$, we aim to find the sparsest coefficient vector $\boldsymbol{w}$ to represent $\boldsymbol{x}$ as a linear combination of basis functions from $\boldsymbol{C}$ by

$$\text{minimising} \quad ||\boldsymbol{w}||_0 \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{C}\boldsymbol{w}\,.$$

In general, dictionaries are designed to be overcomplete, i.e. the number of columns exceeds the number of rows. Minimising the zero-norm requires enumerating all dictionary subsets which is computationally infeasible. Again, a one-norm based approximation is well suited to approximate the otherwise intractable optimisation problem. An equivalence condition has been derived in [Donoho and Elad, 2003] to assess whether the optimal zero-norm solution is found by minimising the one-norm of the coefficient vector.

However, the results do not apply to the SFM as both methods are orthogonal in the way they define the term *sparsity*. The SFM defines sparsity with respect to features, sparse coding defines sparsity with respect to the data representation by basis functions or dictionary entries.

Naming Conflicts Assigning an intuitive, self-explanatory and yet unoccupied name to a novel method is nothing trivial — and so was the choice *support feature machine*. First, the name needs to reflect its essential purpose which is feature selection. Second, the method is obviously inspired by the theory of *support vector machines* but differs in some essential aspects. So, putting a prefix in front of SVM might be misleading — in particular the SFM does not maximise a margin in any sense. Finally, the name should be unique and it should not have been used before for any other method. Unfortunately, we do violate this third requirement. Coincidentally, at least three research groups — including our group — have published at the same time a method called support feature machine. We shortly summarise these methods in chronological order and explain why they claimed the term support feature machine.

The SFM was invented for the first time in [CHAOVALITWONGSE et al., 2007] as a method to study multidimensional time series classification. In particular, the authors proposed a method for detecting abnormal brain activity such as epilepsy based on *electroencephalography* (EEG) data. They aim to incorporate both temporal and spatial data into a single optimisation model. The derived optimisation problem combines information from neighbouring EEG electrodes to build a stronger classifier. Here, the term *support feature* refers to the optimal group of electrodes to distinguish between epilepsy and normal brain activity.

The second SFM-variant [MASZCZYK and DUCH, 2010a,MASZCZYK and DUCH, 2010b] extracts new features from the original data in a canonical way and combines them in a new feature space. They use features derived from a kernel function, i.e. $z_i = K(\boldsymbol{x}, \boldsymbol{x}_i)$, or from linear projections on the connecting line between cluster centres and even from arbitrary projections of the input data. These features altogether are supposed to provide better discriminative power than any of the feature sets alone. Here, the term *support feature* describes the process of feature generation and combination to improve classification performance.

In contrast, our definition of an SFM focuses on feature selection in the sense of dimension reduction. The *support features* constitute the smallest set of features that allows a separation of two classes without error.

3.2 Extensions

Extension to Soft Separability In general, if $n \leq d + 1$, the data is separable and the SFM has a solution. In the following, we introduce slack-variables similar as for soft-margin SVMs to allow for misclassifications during training. This is done for two reasons: First, if the input data is not separable in the intrinsic feature space, i.e. if the classes overlap, irrelevant features will be added to achieve separation of the training data. This leads to an overestimation of the number of truly relevant features and might diminish generalisation performance. Second, even if the

classes are separable in the intrinsic feature space, the true separating hyperplane might not be identified correctly due to outliers. To address these problems, a mechanism is needed that allows for misclassifications and thereby provides a better estimate of the true dimensionality. Note that we do not address the problem of intrinsically non-linear decision borders.

We introduce slack variables ξ_i for each data point and a softness parameter C [Klement and Martinetz, 2010a] in the same way this is done for soft-margin SVMs, i.e. we

$$\begin{aligned} &\text{minimise} && \|\boldsymbol{w}\|_0^0 + C\|\boldsymbol{\xi}\|_0^0 \\ &\text{subject to} && \begin{cases} y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geq -\xi_i \quad \forall i \\ \frac{1}{n}\sum_{i=1}^{n} y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) = \pm 1 \\ \xi_i \geq 0\,. \end{cases} \end{aligned} \tag{3.3}$$

As classification errors are allowed, $y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right)$ may become negative and the pathological case where $\frac{1}{n}\sum_{i=1}^{n} y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right)$ is smaller than zero may occur. Therefore, the optimiser needs to fulfil the latter constraint with +1 or −1. In practise, one needs to optimise for both variants and finally choose the solution with the lower objective function. To solve (3.3), we use the same iterative approximation scheme as described above. An important property of our approach is that the objective function explicitly trades off the number of features $\|\boldsymbol{w}\|_0^0$ and the number of misclassified training samples $\|\boldsymbol{\xi}\|_0^0$.

Extension to Unbalanced Datasets A frequent issue of soft classifiers is their sensitivity to unbalanced datasets. If one class contains more samples than the other, many classifiers tend to behave like a majority classifier and ignore the smaller class. Several solutions to this problem have been proposed such as re-balancing the data artificially by oversampling and undersampling or synthetic sampling, adjusting the output threshold of the classifier according to the data distribution, applying one-class classifiers for one or both classes, and cost-sensitive methods [Provost, 2000, Japkowicz, 2000, Chawla et al., 2004, He and Garcia, 2009].

Our approach is to adjust the softness of the SFM according to the class ratio. We start with an example where the above soft SFM does not provide a valid solution due to class unbalance. Assume a dataset of size n where n^+ samples belong to one class and n^- samples to the other class. In any valid solution the vector $\boldsymbol{w}$ has at least one non-zero entry. Thus, the objective function $\|\boldsymbol{w}\|_0^0 + C\|\boldsymbol{\xi}\|_0^0$ is at least 1 if the data is separable within one dimension, and larger if the slack variables take non-vanishing values. However, if the class sizes differ, all constraints are fulfilled by setting

$$\boldsymbol{w} = \mathbf{0}, \quad b = \frac{n}{n^+ - n^-} \quad \text{and} \quad \xi_i = \left|\frac{n}{n^+ - n^-}\right|$$

because

$$\begin{aligned} y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) &= y_i \frac{n}{n^+ - n^-} \geq - \left| \frac{n}{n^+ - n^-} \right| = -\xi_i \quad \text{and} \\ \frac{1}{n} \sum_{i=1}^{n} y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) &= \frac{1}{n} \left(n^+ \frac{n}{n^+ - n^-} - n^- \frac{n}{n^+ - n^-} \right) = 1 \,. \end{aligned}$$

For $C < \frac{|n^+ - n^-|}{n^2}$, the objective function is even smaller than one for all n^+ and n^- with $n = n^+ + n^-$:

$$\|\boldsymbol{w}\|_0^0 + C \|\boldsymbol{\xi}\|_0^0 < \frac{|n^+ - n^-|}{n^2} \, n = \frac{|n^+ - n^-|}{n} \leq 1 \,.$$

In this case, the optimal solution would have a smaller objective value than any solution with a non-zero weight vector may have. This can be avoided by introducing class-specific softness parameters and by adjusting the equality constraint. For convenience, we define two sets of indices — one for each class — i.e. $I^+ = \{ i \,|\, y_i = +1 \}$ and $I^- = \{ i \,|\, y_i = -1 \}$. Then, we

$$\begin{aligned} &\text{minimise} && \|\boldsymbol{w}\|_0^0 + C^+ \|\boldsymbol{\xi}^+\|_0^0 + C^- \|\boldsymbol{\xi}^-\|_0^0 \\ &\text{subject to} && \begin{cases} y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) \geq -\xi_i^+ & \text{for all } i \in I^+ \\ y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) \geq -\xi_i^- & \text{for all } i \in I^- \\ \frac{1}{n^+} \sum\limits_{i \in I^+} \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) - \frac{1}{n^-} \sum\limits_{i \in I^-} \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) = \pm 1 & \\ \xi_i^+, \xi_i^- \geq 0 \,. & \end{cases} \end{aligned} \tag{3.4}$$

In this formulation, individual misclassification costs can be assigned to each class. In practice, choosing C^+ and C^- such that $C^+ n^+ = C^- n^-$ enforces the proportion of misclassified samples to be equal for both classes. In this setting $\boldsymbol{w} = \boldsymbol{0}$ is avoided, since then

$$\pm 1 = \frac{1}{n^+} \sum_{i \in I^+} b - \frac{1}{n^-} \sum_{i \in I^-} b = b - b = 0 \,.$$

Thus, it is ensured that the solution is non-trivial.

Behaviour in the Limit The extension proposed above reduces the impact of single outliers on the separating hyperplane by trading off the number of obtained features and the number of misclassified samples. To complete the softness extension, we consider the behaviour of the soft SFM in the limit for $C^\pm \to \infty$ and $C^\pm \to 0$. In the first case, the dominant slack term $C^+ \|\boldsymbol{\xi}^+\|_0^0 + C^- \|\boldsymbol{\xi}^-\|_0^0$ forces the slack variables to zero such that we obtain the hard SFM. The opposite case, $C^+ \to 0$, allows arbitrary choices of the slack variables ξ_i^+ and ξ_i^- such that the objective function becomes independent of the misclassification rate. Thus, the inequality

constraints are fulfilled for all $\boldsymbol{w}$ and b. In the limit, the optimisation problem (3.4) simplifies to

$$\begin{aligned} &\text{minimise} && \|\boldsymbol{w}\|_0^0 \\ &\text{subject to} && \tfrac{1}{n^+} \sum_{i \in I^+} \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) - \tfrac{1}{n^-} \sum_{i \in I^-} \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) = 1 \,. \end{aligned}$$

Assume the SFM identifies one and only one feature to be relevant, i.e. the objective value is 1 and the weight vector differs from zero in exactly one entry. Let j be the index of this non-zero entry. Then, the equality constraint is solved with respect to w_j by

$$\begin{aligned} & \frac{1}{n^+} \sum_{i \in I^+} \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) - \frac{1}{n^-} \sum_{i \in I^-} \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) &&= 1 \\ \Leftrightarrow \quad & \frac{1}{n^+} \sum_{i \in I^+} \left(w_j x_{ij} + b \right) - \frac{1}{n^-} \sum_{i \in I^-} \left(w_j x_{ij} + b \right) &&= 1 \\ \Leftrightarrow \quad & w_j \left(\frac{\sum_{i \in I^+} x_{ij}}{n^+} + b - \frac{\sum_{i \in I^-} x_{ij}}{n^-} - b \right) &&= 1 \\ \Leftrightarrow \quad & w_j &&= \frac{1}{\mu_j^+ - \mu_j^-} \end{aligned}$$

where μ_j^+ and μ_j^- are the class specific means. As the SFM approximates the zero-norm by minimising the one-norm, it will select the feature which minimises $|w_j|$ and, therefore, maximises the distance of the class specific means $\left| \mu_j^+ - \mu_j^- \right|$. This is closely related to correlation-based feature selection methods (see Chapter 2.6.3). Thus, we expect the soft SFM to favour those features that maximise the correlation between feature value and class label, or maximise the difference between the feature values of the two classes. In total, the soft SFM is a trade-off between a hard SFM and correlation-based feature ranking.

Repetitive Feature Selection The SFM described so far extracts a single set of relevant features from a high-dimensional dataset. Specifically, the SFM finds the *smallest* set of features in which the two classes are linearly separable. As we will show in Chapter 4.2, this approach identifies truly relevant features with high reliability in many high-dimensional small sample size scenarios, particularly if the data contains few relevant and many irrelevant dimensions. However, high-dimensional real-world datasets often contain *several* informative feature subsets that all permit linear separation. In such scenarios, one might not only be interested to find the most informative features, but also to identify *all* informative features. Although the exact number of features that carry information alone or in combination with others can often not be determined in such datasets — the sample size is usually insufficient to capture all sources of variance and to accurately describe the decision border — the total amount of informative

features might be determined with some simplifications and heuristics. In this section, we propose a way how the SFM can be used to identify both the most informative and the least informative features and to estimate the fraction of informative and uninformative features.

The basic idea of the repetitive SFM approach (see Algorithm 3.3) is to train an SFM on the complete dataset, remove all obtained features from the dataset, retrain on the reduced dataset, discard the obtained features again, retrain again, and so on, until the dataset is no longer separable within the remaining features. If the repetitive SFM (RSFM) correctly identifies the smallest informative feature set in each run, the size of the returned feature subsets will monotonously increase as more and more features are discarded. However, in practise this might not always be the case because the optimisation might terminate in a local optimum due to the dataset configuration or because of numerical issues of the technical implementation of the SFM. To correct for such inaccuracies, we sort the obtained feature subsets according to their size, starting with the smallest feature subset. This way we obtain a sequence of monotonously increasing feature subsets that, according to our definition, represents a sequence of feature subsets which are less and less relevant for classification. The number of informative features can then be estimated depending on (i) the size of the feature subset, (ii) the generalisation error of the RSFM, or (iii) the generalisation error of an SVM trained on the features that remained in the dataset after all features identified by the SFM in a particular repetition had been discarded.

In Chapters 4.4 and 6, we show that the RSFM finds the relevant features (as identified with

Input : Feature vectors $\boldsymbol{x}_i$ and class labels y_i
Output: For every iteration t, a weight vector $\boldsymbol{w}_t$, a bias b_t, a set of active features $\mathcal{F}_t$ and a set of relevant features $\mathcal{R}_t$

Initialise the set of active features $\mathcal{F}_0 \leftarrow \{1, \ldots, d\}$
Set $t \leftarrow 0$
repeat
    Train a support feature machine using the feature set $\mathcal{F}_t$
    if *a solution was found* **then**
        Store the results, i.e. $\boldsymbol{w}_t$ and b_t
        Store the set of relevant features, i.e. $\mathcal{R}_t = \{i \,|w_{t,i} \neq 0\}$
        Update the set of relevant features, i.e. $\mathcal{F}_{t+1} = \mathcal{F}_t \setminus \mathcal{R}_t$
        Reduce all feature vectors to $\mathcal{F}_{t+1}$
    end
until *the data is no longer separable within the remaining features*

Figure 3.3: Repetitive support feature machine.

standard univariate approaches) of real-world microarray and neuroimaging datasets with high accuracy and we discuss how a slightly altered version of the RSFM might be used to derive an estimate of the total number of informative features within a given dataset.

3.3 Mathematical Considerations

The support feature machine enforces linear separation with an additional constraint on the mean decision value to avoid the trivial solution $\boldsymbol{w} = \mathbf{0}$. However, it is not obvious why this novel method should be better suited to identify the minimum number of relevant features than, e.g. the closely related method by [WESTON et al., 2003]. Both methods obtain a local minimum of the target problem (2.8). As our simulations show (see Chapter 4.2), the SFM finds the relevant features basically with the first step. WESTON's approach can hardly catch up with the following iterations. It seems that the first step is important and decides, whether we will converge into a good minimum. The first step is equivalent to minimising the one-norm instead of the zero-norm. We derive necessary conditions for finding a zero-norm solution by minimising the one-norm (see also [KLEMENT and MARTINETZ, 2011]) — both for the SFM and the related SVM-based method by WESTON et al. Based on this condition, we explain why the SFM approach finds a zero-norm minimising solution more frequently by comparing their behaviour in a simple illustrative scenario.

First, we introduce some simplifications and notations to improve the readability of the admittedly complex plausibility considerations. In the following, we assume the dataset $\mathcal{D}$ to be linearly separable *without bias*, i.e.

$$\exists\, \boldsymbol{w} \in \mathbb{R}^d \quad \text{with} \quad y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{w} \geq 0 \;\; \forall\, i \quad \text{and} \quad \boldsymbol{w} \neq \mathbf{0}\,,$$

where the normal vector $\boldsymbol{w} \in \mathbb{R}^d$ describes the separating hyperplane except for a constant factor. For abbreviation, we use $\boldsymbol{z}_i = y_i \boldsymbol{x}_i$ and $\boldsymbol{Z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n)$ and $\overline{\boldsymbol{z}} = \sum_{i=1}^{n} \boldsymbol{z}_i$. Thus, WESTON et al. aim to

$$\text{minimise} \quad ||\boldsymbol{w}||_0 \quad \text{subject to} \quad \boldsymbol{Z}^{\mathrm{T}} \boldsymbol{w} \geq \mathbf{1}\,, \tag{3.5}$$

while in the SFM setting we aim to

$$\text{minimise} \quad ||\boldsymbol{w}||_0 \quad \text{subject to} \quad \boldsymbol{Z}^{\mathrm{T}} \boldsymbol{w} \geq \mathbf{0} \quad \text{and} \quad \overline{\boldsymbol{z}}^{\mathrm{T}} \boldsymbol{w} = 1\,. \tag{3.6}$$

Both, (3.5) and (3.6), solve (2.8), but which setting does it more effectively within the first

iteration? For both approaches, we replace the zero-norm by the one-norm such that we

$$\text{minimise} \quad ||\boldsymbol{w}||_1 \quad \text{subject to} \quad \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{w} \geq \mathbf{1}\,, \tag{3.7}$$

for Weston's approach and we

$$\text{minimise} \quad ||\boldsymbol{w}||_1 \quad \text{subject to} \quad \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{w} \geq \mathbf{0} \quad \text{and} \quad \overline{\boldsymbol{z}}^{\mathrm{T}}\boldsymbol{w} = 1 \tag{3.8}$$

in case of the SFM. First, we focus on (3.5). When do we find a solution to (3.5) by solving (3.7)? We denote the solution space of (3.5) by Ω and define the following two weight vectors:

$$\boldsymbol{w}_0 = \underset{\boldsymbol{w}\in\Omega}{\arg\min} \quad ||\boldsymbol{w}||_1 \quad \text{subject to} \quad \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{w} \geq \mathbf{1} \tag{3.9}$$

$$\boldsymbol{w}_1 = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\arg\min} \quad ||\boldsymbol{w}||_1 \quad \text{subject to} \quad \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{w} \geq \mathbf{1} \tag{3.10}$$

Here, $||\boldsymbol{w}_0||_0 = k$, i.e. at least k features are necessary to separate the input data. In the following, we assume $\boldsymbol{w}_0$ and $\boldsymbol{w}_1$ to be unique. Non-uniqueness will occur only in degenerate cases, and since $\boldsymbol{Z}$ is drawn from a probability distribution, the probability of these cases is of measure zero. The probabilistic nature of the input data also ensures that all quadratic submatrices of $\boldsymbol{Z}$ have full rank. Among all solutions of (3.5), $\boldsymbol{w}_0$ is the solution with lowest one-norm. Note that if $\boldsymbol{w}_1$ is in Ω then $\boldsymbol{w}_1 = \boldsymbol{w}_0$. Since in practise, (3.5) cannot be solved directly, Ω is in general unknown as well as $\boldsymbol{w}_0$. However, both are well-defined. In contrast, $\boldsymbol{w}_1$ is the solution on the entire $\mathbb{R}^d$ and can efficiently be found by linear programming. If $\boldsymbol{w}_0 = \boldsymbol{w}_1$ for a specific dataset, then the optimal feature set can be obtained by optimising for the one-norm. Without loss of generality, for the following considerations we assume:

1. All entries of the weight vector are positive, i.e. $w_{0,i} \geq 0$. Otherwise, we invert the corresponding input dimension.

2. The training data is ordered such that $\boldsymbol{Z} = \left(\hat{\boldsymbol{Z}}\ \check{\boldsymbol{Z}}\right)$ with $\hat{\boldsymbol{Z}}^{\mathrm{T}}\boldsymbol{w}_0 = \mathbf{1}$ and $\check{\boldsymbol{Z}}^{\mathrm{T}}\boldsymbol{w}_0 > \mathbf{1}$. So, only the first columns of the design matrix $\boldsymbol{Z}$ correspond to active constraints, i.e. the constraints are fulfilled with equality. Let k^* be the number of active constraints.

3. The dimensions of $\mathcal{D}$ are sorted, such that the first k dimensions of $\boldsymbol{w}_0$ are non-zero:

$$w_{0,i}\begin{cases} > 0 & i = 1,\ldots,k \\ = 0 & \text{otherwise} \end{cases} \qquad \text{such that} \quad \boldsymbol{w}_0 = \begin{pmatrix} \hat{\boldsymbol{w}}_0 \\ \mathbf{0} \end{pmatrix}$$

.

In total, the design matrix $\boldsymbol{Z}$ is organised as

$$\boldsymbol{Z} = \begin{pmatrix} \hat{\boldsymbol{Z}}_1 & \check{\boldsymbol{Z}} \\ \hat{\boldsymbol{Z}}_2 & \end{pmatrix} \quad \text{with} \quad \hat{\boldsymbol{Z}}_1 \in \mathbb{R}^{k \times k^*}, \hat{\boldsymbol{Z}}_2 \in \mathbb{R}^{d-k \times k^*}, \check{\boldsymbol{Z}} \in \mathbb{R}^{d \times n-k^*}$$

where the dimensions n and d are known in advance. The following lemma holds for the relation between k and k^*:

Lemma 3.3.1 *If $\boldsymbol{w}_0$ contains k non-zero entries, exactly k constraints in $\boldsymbol{Z}^T\boldsymbol{w}_0 \geq \mathbf{1}$ are active, i.e. $k = k^*$.*

Proof In linear programming theory, a *basic feasible solution* is defined to be a solution located in one of the corners of the solution space defined by the constraints. The *fundamental theorem of linear programming* (to be found in many textbooks on linear programming, e.g. in [Dantzig and Thapa, 2003, Vanderbei, 2008]) states that if an optimal solution exists, then also a *basic optimal solution* exists. In other words, optimal solutions are located in the corners of the solution space, which is exploited by the *simplex method* for solving linear programming problems.

As stated before, (3.5) may have multiple solutions. Each solution may involve a different set of features. Let Λ_i be the linear subspace spanned by the k features of a particular solution to (3.5). Then, $\Omega \subset \bigcup_i \Lambda_i$, and

$$\begin{aligned} \boldsymbol{w}_0 &= \underset{\boldsymbol{w} \in \Omega}{\arg\min} \quad ||\boldsymbol{w}||_1 \quad \text{subject to} \quad \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{w} \geq \mathbf{1} \\ &= \underset{\boldsymbol{w} \in \bigcup_i \Lambda_i}{\arg\min} \quad ||\boldsymbol{w}||_1 \quad \text{subject to} \quad \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{w} \geq \mathbf{1} \end{aligned}$$

is valid. Thus, $\boldsymbol{w}_0$ can be obtained by a sequence of linear programs. All of them are feasible and non-degenerate. Therefore, an optimal solution exists, and — according to the fundamental theorem of linear programming — a basic optimal solution can be found. By definition, $\boldsymbol{w}_0$ contains k non-zeroentries, so $\hat{\boldsymbol{Z}}^{\mathrm{T}}\boldsymbol{w}_0 = \hat{\boldsymbol{Z}}_1^{\mathrm{T}}\hat{\boldsymbol{w}}_0 = \mathbf{1}$, i.e. the initial d-dimensional problem is equivalent to a k-dimensional one. In a basic solution, k constraints are active and, hence, $k^* = k$ follows. □

We proceed with the main theorem, that provides a necessary condition for finding $\boldsymbol{w}_0$ with a linear program, i.e. by solving (3.10).

Theorem 3.3.2 *For $\boldsymbol{w}_1 = \boldsymbol{w}_0$, it is necessary that* $\left\| \hat{\boldsymbol{Z}}_2 \hat{\boldsymbol{Z}}_1^{\mathrm{T}} \left(\hat{\boldsymbol{Z}}_1 \hat{\boldsymbol{Z}}_1^{\mathrm{T}} \right)^{-1} \mathbf{1} \right\|_\infty < 1.$

Proof If $\boldsymbol{w}_0 = \boldsymbol{w}_1$, for each infinitesimal disparity vector $\boldsymbol{\Delta}$ with $\hat{\boldsymbol{Z}}^{\mathrm{T}}(\boldsymbol{w}_0 + \boldsymbol{\Delta}) = \mathbf{1}$ and $\check{\boldsymbol{Z}}^{\mathrm{T}}(\boldsymbol{w}_0 + \boldsymbol{\Delta}) > \mathbf{1}$ we have

$$\begin{aligned}
&||\boldsymbol{w}_0 + \boldsymbol{\Delta}||_1 &&> &&||\boldsymbol{w}_0||_1 \\
\Leftrightarrow\quad &\sum_{i=1}^{d} |w_{0,i} + \Delta_i| &&> &&\sum_{i=1}^{d} |w_{0,i}| = \sum_{i=1}^{d} w_{0,i} \\
\Leftrightarrow\quad &\sum_{i=1}^{k} |w_{0,i} + \Delta_i| + \sum_{i=k+1}^{d} |\underbrace{w_{0,i}}_{=0} + \Delta_i| &&> &&\sum_{i=1}^{k} w_{0,i} \\
\Leftrightarrow\quad &\sum_{i=1}^{k} w_{0,i} + \Delta_i + \sum_{i=k+1}^{d} |\Delta_i| &&> &&\sum_{i=1}^{k} w_{0,i} \\
\Leftrightarrow\quad &\sum_{i=1}^{k} \Delta_i + \sum_{i=k+1}^{d} |\Delta_i| &&> &&0\,. \qquad (3.11)
\end{aligned}$$

Next, we make use of the particular structure of the matrix $\hat{\boldsymbol{Z}}$ and split the disparity vector into an upper and a lower part, i.e. $\boldsymbol{\Delta}^{\mathrm{T}} = \left(\boldsymbol{\Delta}_1^{\mathrm{T}} \boldsymbol{\Delta}_2^{\mathrm{T}}\right)$ with $\boldsymbol{\Delta}_1 \in \mathbb{R}^k$, $\boldsymbol{\Delta}_2 \in \mathbb{R}^{d-k}$. A closed formulation for $\boldsymbol{\Delta}_1$ is derived by rearrangement and using $\hat{\boldsymbol{Z}}^{\mathrm{T}} \boldsymbol{\Delta} = \mathbf{0}$:

$$\begin{aligned}
\hat{\boldsymbol{Z}}^{\mathrm{T}} \boldsymbol{\Delta} = \hat{\boldsymbol{Z}}_1^{\mathrm{T}} \boldsymbol{\Delta}_1 + \hat{\boldsymbol{Z}}_2^{\mathrm{T}} \boldsymbol{\Delta}_2 &= \mathbf{0} \\
\Leftrightarrow\quad \hat{\boldsymbol{Z}}_1^{\mathrm{T}} \boldsymbol{\Delta}_1 &= -\hat{\boldsymbol{Z}}_2^{\mathrm{T}} \boldsymbol{\Delta}_2 \\
\Leftrightarrow\quad \hat{\boldsymbol{Z}}_1 \hat{\boldsymbol{Z}}_1^{\mathrm{T}} \boldsymbol{\Delta}_1 &= -\hat{\boldsymbol{Z}}_1 \hat{\boldsymbol{Z}}_2^{\mathrm{T}} \boldsymbol{\Delta}_2 \\
\Leftrightarrow\quad \boldsymbol{\Delta}_1 &= -\left(\hat{\boldsymbol{Z}}_1 \hat{\boldsymbol{Z}}_1^{\mathrm{T}}\right)^{-1} \hat{\boldsymbol{Z}}_1 \hat{\boldsymbol{Z}}_2^{\mathrm{T}} \boldsymbol{\Delta}_2 \qquad (3.12) \\
\Rightarrow\quad \mathbf{1}^{\mathrm{T}} \boldsymbol{\Delta}_1 &= -\underbrace{\mathbf{1}^{\mathrm{T}} \left(\hat{\boldsymbol{Z}}_1 \hat{\boldsymbol{Z}}_1^{\mathrm{T}}\right)^{-1} \hat{\boldsymbol{Z}}_1 \hat{\boldsymbol{Z}}_2^{\mathrm{T}}}_{:=\boldsymbol{\alpha}^{\mathrm{T}}} \boldsymbol{\Delta}_2\,.
\end{aligned}$$

Finally, (3.11) can be expressed using $\boldsymbol{\alpha}$ and $\boldsymbol{\Delta}_2$:

$$\sum_{i=1}^{k} \Delta_i + \sum_{i=k+1}^{d} |\Delta_i| = -\boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{\Delta}_2 + ||\boldsymbol{\Delta}_2||_1 = \sum_{i=k+1}^{d} -\alpha_{i-k} \Delta_i + |\Delta_i| \quad > \quad 0\,. \qquad (3.13)$$

Equation (3.13) has to hold for any infinitesimal $\boldsymbol{\Delta}_2$, which is the case if and only if $|\alpha_{i-k}| < 1$ holds for all $k + 1 \leq i \leq d$, i.e. if and only if

$$||\boldsymbol{\alpha}||_\infty = \left\| \hat{\boldsymbol{Z}}_2 \hat{\boldsymbol{Z}}_1^{\mathrm{T}} \left(\hat{\boldsymbol{Z}}_1 \hat{\boldsymbol{Z}}_1^{\mathrm{T}}\right)^{-1} \mathbf{1} \right\|_\infty < 1. \qquad (3.14)$$

(Note: $\boldsymbol{\Delta}_2 = \mathbf{0}$ is excluded according to (3.12), since then $\boldsymbol{\Delta} = \mathbf{0}$). □

So far, all considerations apply for the optimisation problem (3.5). However, with the following minor changes a similar condition can be derived for the SFM-problem (3.6):

1. The design matrix $\boldsymbol{Z}$ is extended by an additional column, the vector $\overline{\boldsymbol{z}}$.

2. The weight vectors $\boldsymbol{w}_0$ and $\boldsymbol{w}_1$ are defined analogously:

$$\begin{aligned} \boldsymbol{w}_0 &= \underset{\boldsymbol{w}\in\Omega}{\arg\min} \quad \|\boldsymbol{w}\|_1 \quad \text{subject to} \quad (\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n)^{\mathrm{T}}\boldsymbol{w} \geq \boldsymbol{0} \quad \text{and} \quad \overline{\boldsymbol{z}}^{\mathrm{T}}\boldsymbol{w} = 1 \\ \boldsymbol{w}_1 &= \underset{\boldsymbol{w}\in\mathbb{R}^d}{\arg\min} \quad \|\boldsymbol{w}\|_1 \quad \text{subject to} \quad (\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n)^{\mathrm{T}}\boldsymbol{w} \geq \boldsymbol{0} \quad \text{and} \quad \overline{\boldsymbol{z}}^{\mathrm{T}}\boldsymbol{w} = 1 \end{aligned}$$

3. If $\boldsymbol{w}_0$ contains k non-zero entries, exactly k constraints are active. The last of these constraints is the equality constraint $\overline{\boldsymbol{z}}^{\mathrm{T}}\boldsymbol{w} = 1$.

4. The design matrix $\boldsymbol{Z}$ and the weight vector $\boldsymbol{w}_0$ are ordered in the same way as before, i.e. the first k entries of $\boldsymbol{w}_0$ are non-zero and the first k columns of $\boldsymbol{Z}$ correspond to active constraints. The kth column of $\boldsymbol{Z}$ contains $\overline{\boldsymbol{z}}$.

Theorem 3.3.2 and its proof now apply exactly in the same way also to the SFM. Both approaches are very closely connected, but the slight difference leads to a significantly lower number of features for the SFM. It is not possible to give a rigorous mathematical proof for the superior performance of the SFM (3.6) compared to WESTON's approach (3.5) in general. However, within a simplified scenario and with approximate arguments we can use the result of the above theorem to make the superior performance plausible. Therefore, we consider the most simple scenario. Assume the elements of each vector $\boldsymbol{z}_i$ to be drawn from a normal distribution $\mathcal{N}(\mu,\sigma^2)$ with the expected value

$$\mu = \begin{cases} c & i = 1,\ldots,k,\ \ c \neq 0 \\ 0 & \text{otherwise}\,. \end{cases}$$

Thus, only the first k features are relevant and all others are irrelevant. For WESTON's approach (3.5) we have $\hat{\boldsymbol{Z}}_1^{\mathrm{T}}\hat{\boldsymbol{w}}_0 = \boldsymbol{1}$ and obtain

$$\hat{\boldsymbol{Z}}_1\hat{\boldsymbol{Z}}_1^{\mathrm{T}}\hat{\boldsymbol{w}}_0 = \hat{\boldsymbol{Z}}_1\boldsymbol{1} \approx k\cdot c\cdot\boldsymbol{1} \qquad \Leftrightarrow \qquad \hat{\boldsymbol{w}}_0 \approx k\cdot c\cdot\left(\hat{\boldsymbol{Z}}_1\hat{\boldsymbol{Z}}_1^{\mathrm{T}}\right)^{-1}\boldsymbol{1}$$

such that

$$\|\boldsymbol{\alpha}\|_\infty = \left\|\hat{\boldsymbol{Z}}_2\hat{\boldsymbol{Z}}_1^{\mathrm{T}}\left(\hat{\boldsymbol{Z}}_1\hat{\boldsymbol{Z}}_1^{\mathrm{T}}\right)^{-1}\boldsymbol{1}\right\|_\infty \approx \left\|\frac{\hat{\boldsymbol{Z}}_2\hat{\boldsymbol{Z}}_1^{\mathrm{T}}\hat{\boldsymbol{w}}_0}{k\cdot c}\right\|_\infty = \left\|\frac{\hat{\boldsymbol{Z}}_2\boldsymbol{1}}{k\cdot c}\right\|_\infty = \left\|\frac{\boldsymbol{\varepsilon}_k}{c}\right\|_\infty .$$

The entries of the vector $\boldsymbol{\varepsilon}_k$ are distributed as $\mathcal{N}(0, \sigma^2/k)$. In contrast, for the SFM (3.6), where the last column of $\hat{\boldsymbol{Z}}$ is the mean of all $\boldsymbol{z}_i$, we have $\hat{\boldsymbol{Z}}_1^{\mathrm{T}} \hat{\boldsymbol{w}}_0 = \binom{\mathbf{0}}{1}$ and obtain

$$\hat{\boldsymbol{Z}}_1 \hat{\boldsymbol{Z}}_1^{\mathrm{T}} \hat{\boldsymbol{w}}_0 = \hat{\boldsymbol{Z}}_1 \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \approx c \cdot \mathbf{1} \qquad \Leftrightarrow \qquad \hat{\boldsymbol{w}}_0 \approx c \cdot \left(\hat{\boldsymbol{Z}}_1 \hat{\boldsymbol{Z}}_1^{\mathrm{T}} \right)^{-1} \mathbf{1}$$

and

$$\|\boldsymbol{\alpha}\|_\infty \approx \left\| \frac{\hat{\boldsymbol{Z}}_2 \hat{\boldsymbol{Z}}_1^{\mathrm{T}} \hat{\boldsymbol{w}}_0}{c} \right\|_\infty = \left\| \frac{\hat{\boldsymbol{Z}}_2}{c} \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \right\|_\infty = \left\| \frac{\boldsymbol{\varepsilon}_n}{c} \right\|_\infty .$$

Here, the entries of $\boldsymbol{\varepsilon}_n$ are distributed as $\mathcal{N}(0, \sigma^2/n)$. Obviously, for $k \ll n$, the probability that all elements of $\boldsymbol{\alpha}$ stay below 1 and, hence, that the condition in Theorem 3.3.2 is fulfilled, is much larger for the SFM. As expected, the larger c, the easier it is for both approaches to be successful. Note, that we assumed that the elements of $\hat{\boldsymbol{Z}}_1$ and $\hat{\boldsymbol{Z}}_2$ are independent stochastic variables. Of course, since $\hat{\boldsymbol{Z}}_1$ and $\hat{\boldsymbol{Z}}_2$ are selected by the respective algorithm according to certain criteria, this is not necessarily the case.

To summarise, the above considerations are no proof for a superior performance of the SFM in general, however, it provides some insight why we observe it to identify relevant features more effectively than the SVM-based feature selection method by WESTON et al. in many scenarios.

3.4 On the VC-Dimension of the Support Feature Machine

The VC-dimension is a measure for the expressive power of a family of classification functions $\mathcal{F}$. Feature selection methods aim to reduce the VC-dimension by restricting the family $\mathcal{F}$ to some subset. In general, the VC-dimension of a particular method cannot be given in closed form, however, with some assumptions, we may derive upper bounds to quantify the reduction of the expressive power. Additionally, as numerous estimates on the generalisation error depend on the VC-dimension, we may also derive limits on the probability to classify a novel sample correctly. In the following, we assume $\mathcal{F}$ to be the family of linear classification functions passing through the origin. Further, assume the inducer $\mathcal{I}$ to select for any given dataset exactly that function that uses the least number of features while classifying all training samples correctly.

Feature selection methods can implement the concept of structural risk minimisation by defining a nested set of classification functions to reduce the initial set of d dimensions. A combined feature selection and classification method derives a classifier that involves only a subset of features, e.g. of size d^*. Naïvely, one would assume this classifier to have the same VC-dimension as a d^*-dimensional classifier without feature selection. However, this is not the case as the inducer chooses a specific subset from all d^*-dimensional feature subsets. Thus, the expressive power of the combined feature selection and classification procedure is always

larger than that of the core classifier. Thus, any embedded feature selection method may indeed reduce the dimensionality significantly, however, its VC-dimension is still larger as it would be if the discarded features would not have been present in the input data at all. Consequently, one should avoid to include input features that are known to have no relevance due to prior knowledge whenever possible.

For one-dimensional feature subsets, the VC-dimension of combined feature selection and classification can be derived explicitly. Assume the training samples to be separable within a single out of d dimensions. In this case, the family of classification functions $\mathcal{F}$ consists of $2d$ distinct functions, each corresponding to a one-dimensional classifier either pointing towards positive or negative values. Thus, the growth function is $G_{\mathcal{F}}(d) = 2d$. The VC-dimension h is defined to be the largest sample size such that a configuration exists for which $\mathcal{F}$ shatters the dataset, i.e. allows to classify all 2^h possible labellings correctly. Thus, in case of dimensionality reduction to a single dimension, the VC-dimension is

$$2d = 2^h \quad \Rightarrow \quad h = 1 + \lfloor \log_2 d \rfloor.$$

Thus, the freedom to choose a one-dimensional classifier out of d different ones comes with an increase of the VC-dimension by $\lfloor \log_2 d \rfloor$.

But how does a d-dimensional dataset with $n = h$ data points look like, if it is separable within a single dimension, no matter how we assign the labels? For simplicity, we assume $d = 2^k$, and we restrict all feature values to be either +1 or −1. Thus, we need to choose

$$n = h = 1 + \lfloor \log_2 d \rfloor = 1 + \lfloor \log_2 2^k \rfloor = 1 + k$$

samples from $\{-1, +1\}^{2^k}$ such that for all $2^n = 2^{1+k}$ different labellings a single dimension exists in which the data is separable. A suitable choice would be:

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_n^{\mathrm{T}} \end{pmatrix} \quad \text{with} \quad x_{i,j} = \begin{cases} +1 & \text{if the } j\text{th digit in the binary representation of } i \text{ is } 0 \\ -1 & \text{otherwise.} \end{cases}$$

In Figure 3.4, this data scenario is shown for $d = 2$ ($k = 1$, $n = h = 2$) in connection with all label combinations and the obtained classification functions for $d = 2$. Assume an arbitrary label vector $\boldsymbol{y} \in \{-1, +1\}^d$. Then a dimension j exists with either $(x_{1,j} \dots x_{n,j}) = \boldsymbol{y}$ or $(x_{1,j} \dots x_{n,j}) = -\boldsymbol{y}$. Thus, a classification function taken from $\mathcal{F}$ exists that separates the data in dimension i, i.e. $\mathcal{F}$ shatters the above dataset.

In general, the input data is not separable in just one dimension, but in $1 \leq d^* \leq d$ dimen-

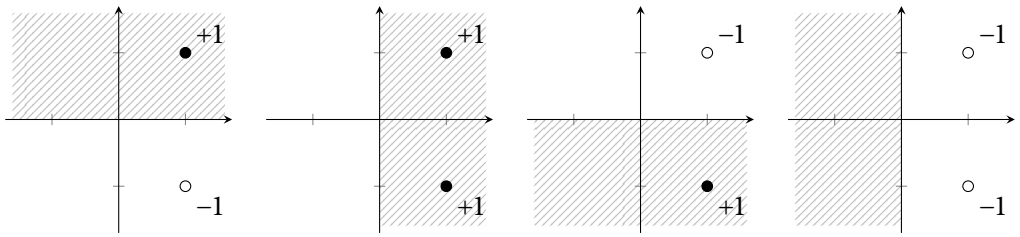


Figure 3.4: Data configuration that is shattered by the family of linear classification functions without bias involving only one feature. Data points from class +1 are shown as black dots (●), while patterns from class −1 are drawn as white dots (○). For any labelling of the input data ($\boldsymbol{x}_1 = \binom{1}{1}$ and $\boldsymbol{x}_2 = \binom{1}{-1}$) a 1-dimensional classifier exists that discriminates all patterns correctly. For each labelling, the corresponding classifier is visualised by marking the region where it classifies data points to belong to class +1 (▨).

sions. Thus, an optimal inducer — one that indeed finds a d^*-dimensional subspace given the d-dimensional input data — has to choose a decision function from a total of $\binom{d}{d^*}$ possible families of classification functions. Each family contains the decision functions restricted to one particular d^*-dimensional subspace. Given two arbitrary families of classification functions $\mathcal{F}_1$ and $\mathcal{F}_2$, the following relation holds for their growth functions:

$$G_{\mathcal{F}_1 \cup \mathcal{F}_2}(n) \leq G_{\mathcal{F}_1}(n) + G_{\mathcal{F}_2}(n)\,.$$

Thus, the growth function $G_{\binom{d}{d^*}}(n)$ of an inducer that combines feature selection and classification in d^* out of d dimensions is upper bounded by

$$G_{\binom{d}{d^*}}(n) \leq \binom{d}{d^*} G_{d^*}(n)$$

where $G_{d^*}(n)$ denotes the growth function of a classifier without feature selection involving d^* dimensions. The growth function of a linear classifier without bias is known to be

$$G_d(n) = \left\{ \begin{array}{ll} 2^n & \text{if } n < d \\ 2\sum_{k=0}^{d-1} \binom{n-1}{k} & \text{otherwise} \end{array} \right\},$$

a fact that has independently been proven by different authors [Sauer, 1972, Wendel, 1962,

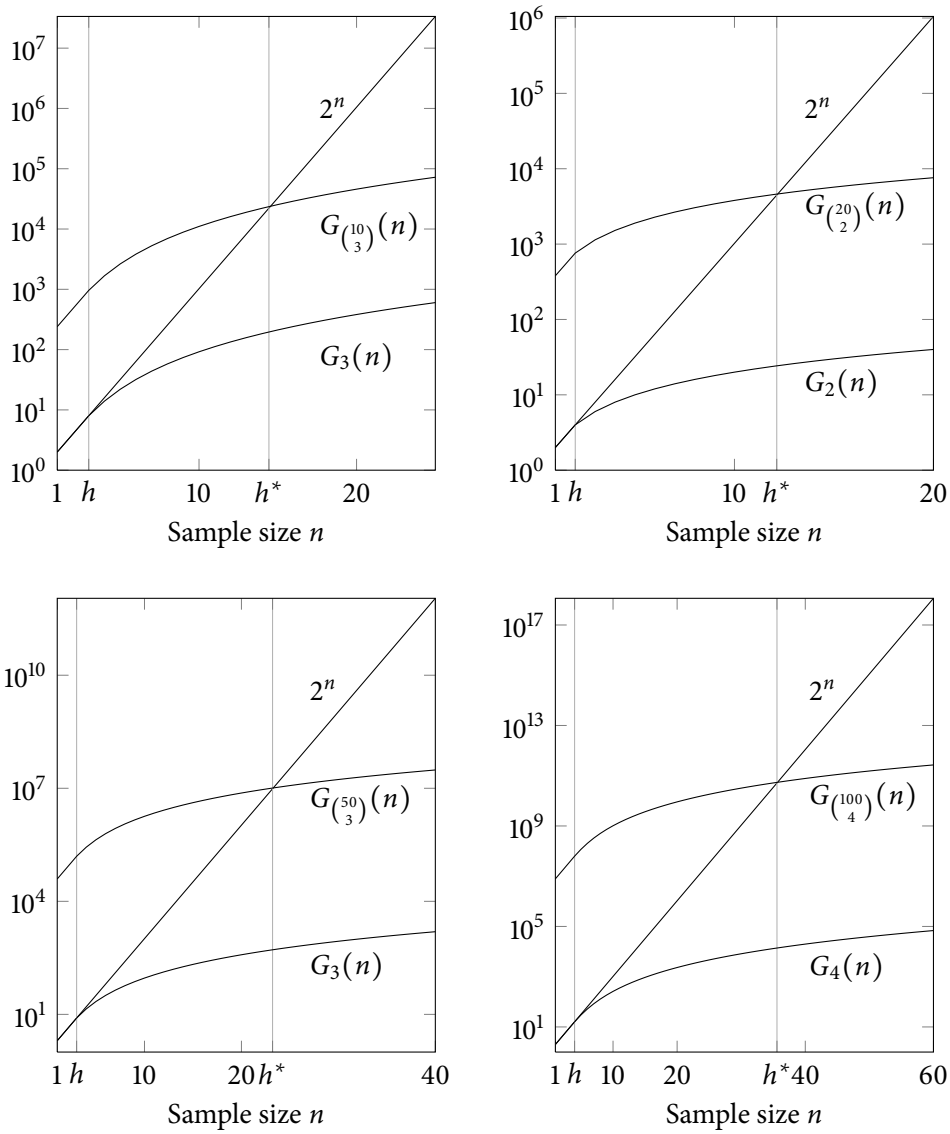


Figure 3.5: Upper bounds of the growth function and the VC-dimension in combined feature selection and classification. Shown are the growth function G_{d^*} of a d^*-dimensional linear classifier, the upper bound on the growth function $G_{\binom{d}{d^*}}$ for combined feature selection and classification when using d^* out of d dimensions, and the respective VC-dimensions h and h^*. The upper bound on the growth function is very rough, so h^* — derived in this way — may be much larger than the true VC-dimension and even larger than d (top left).

Cover, 1965, MacKay, 2002]. With $n \geq d$, the VC-dimension h is upper bounded by h^* with

$$2^{h^*} = \binom{d}{d^*} 2 \sum_{k=0}^{d^*-1} \binom{h^*-1}{k}.$$

Solutions with respect to h^* can easily be derived numerically. In Figure 3.5, the characteristics of 2^n, $G_{d^*}(n)$ and $\binom{d}{d^*} G_{d^*}(n)$ are shown for selected scenarios. Being a very conservative estimate, in some cases h^* may become even larger than h_d — the VC-dimension of a classifier involving all features.

3.5 Implementation using Linear Programming Solvers

So far, the theoretical aspects of the SFM and related feature selection and classification methods have been discussed in detail. Next, we show, how these mathematical formulations are transformed to be solved by standard optimisation frameworks. Given a particular optimisation problem, it is in general impossible to say beforehand which of the numerous commercially or freely available solvers will perform best. Therefore, we chose four optimisation packages to empirically analyse their performance in solving the SFM. Of course, this list is only a small excerpt of the confusingly vast world of linear programming toolboxes, however, they cover the main concepts and algorithms including the simplex algorithm, interior-point methods and presolvers.

We applied two proprietary optimisation packages (CPLEX and MOSEK), one solver included in the numerical computing environment MATLAB, and, finally, the open-source package GLPK. First, we transform the SFM formulation into a standard linear program to be solved using any of the above packages and derive two alternative formulations that differ in size and sparsity of the constraint matrices. Depending on the problem size — the dimension and the sample size — either of the alternatives may be better suited. The aforementioned linear programming solvers (see Figure 3.6) differ slightly in the way the linear program has to be defined — either the equality and inequality constraints are handled separately or they are combined. All solvers provide a MATLAB interface and allow for comprehensive parameter tuning.

Transformation The iterative SFM algorithm (3.2) requires to

$$\begin{aligned} \text{minimise} \quad & |\boldsymbol{w}|_1 \\ \text{subject to} \quad & y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) \geq 0 \quad \text{for all } i \\ \text{and} \quad & \tfrac{1}{n} \textstyle\sum_{i=1}^{n} y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) = 1 \end{aligned} \tag{3.15}$$

Toolbox	Notation	Remarks
CPLEX	The function *cplexlp* minimises $\boldsymbol{f}^{\mathrm{T}}\boldsymbol{x}$ subject to $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ $\boldsymbol{A}^{\mathrm{eq}}\boldsymbol{x} = \boldsymbol{b}^{\mathrm{eq}}$ $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$ with the *objective function* $\boldsymbol{f}$, the *equality constraints* defined by $\boldsymbol{A}$ and $\boldsymbol{b}$, the *inequality constraints* defined by $\boldsymbol{A}^{\mathrm{eq}}$ and $\boldsymbol{b}^{\mathrm{eq}}$ and the *variable bounds* $\boldsymbol{l}$ and $\boldsymbol{u}$.	General purpose optimisation package; distributed as IBM ILOG CPLEX Optimization Studio 12.3; provided free of charge for academic non-commercial use.
MOSEK	The function *msklpopt* minimises $\boldsymbol{f}^{\mathrm{T}}\boldsymbol{x}$ subject to $\boldsymbol{l}^{c} \leq \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{u}^{c}$ $\boldsymbol{l}^{x} \leq \boldsymbol{x} \leq \boldsymbol{u}^{x}$ with the *constraint bounds* $\boldsymbol{l}^{c}$ and $\boldsymbol{u}^{c}$ and the *variable bounds* $\boldsymbol{l}^{x}$ and $\boldsymbol{u}^{x}$.	General purpose toolbox; distributed under commercial licence by MOSEK Aps; academic licences are available free of charge; mainly based on the work of Andersen, e.g. [Andersen and Andersen, 2000].
MATLAB	The function *linprog* minimises $\boldsymbol{f}^{\mathrm{T}}\boldsymbol{x}$ subject to $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ $\boldsymbol{A}^{\mathrm{eq}}\boldsymbol{x} = \boldsymbol{b}^{\mathrm{eq}}$ $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$.	Included in the numerical computing environment MATLAB; developed by MathWorks Inc.; based on the predictor-corrector algorithm [Mehrotra, 1992].
GLPK	The function *glpsol* minimises $\boldsymbol{f}^{\mathrm{T}}\boldsymbol{x}$ subject to $\boldsymbol{l}^{c} \leq \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{u}^{c}$ $\boldsymbol{l}^{x} \leq \boldsymbol{x} \leq \boldsymbol{u}^{x}$.	Freely available open-source package for linear programming and mixed integer programming.

Figure 3.6: Notations of the evaluated linear programming toolboxes.

multiple times. In order to be solved with standard linear programming frameworks, this non-linear optimisation problem needs to be transformed into a linear one. The basic idea is to split each entry of the weight vector into a positive and a negative component where only one may be active, i.e. $w_i = w_i^+ - w_i^-$ with $w_i^+, w_i^- \geq 0$ and $w_i^+ \cdot w_i^- = 0$. As either w_i^+ or w_i^- or both are 0, $|w_i| = w_i^+ + w_i^-$ holds. Thus, we

$$\begin{aligned} &\text{minimise} && \sum_{i=1}^{d} w_i^+ + w_i^- \\ &\text{subject to} && \begin{cases} y_i \left(\sum_{j=1}^{d} \left(w_j^+ - w_j^- \right) x_{ij} + b \right) \geq 0 \quad \text{for all } i \\ \frac{1}{n} \sum_{i=1}^{n} y_i \left(\sum_{j=1}^{d} \left(w_j^+ - w_j^- \right) x_{ij} + b \right) = 1 \\ w_i^+, w_i^- \geq 0 \quad \text{for all } i \ . \end{cases} \end{aligned} \tag{3.16}$$

Note, the constraint $w_i^+ \cdot w_i^- = 0$ is not required: Assume the optimal solution is found and both variables take positive values. Then, one could reduce each of them by $\min(w_i^+, w_i^-)$ without affecting w_i, i.e. the overall weight vector stays the same. However, the objective function is reduced by $2 \cdot \min(w_i^+, w_i^-)$, which is a contradiction to the initial assumption of w_i^+ and w_i^- being optimal. In case of MATLAB's linprog, the input matrices and vectors for the above linear program take the following values:

$$\boldsymbol{f} = \begin{pmatrix} \mathbf{1}^{\mathrm{T}} & \mathbf{1}^{\mathrm{T}} & 0 \end{pmatrix}^{\mathrm{T}} \in \mathbb{R}^{2d+1} \qquad \text{(objective function)}$$

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{w}^{+\mathrm{T}} & \boldsymbol{w}^{-\mathrm{T}} & b \end{pmatrix} \qquad \text{(target variable)}$$

$$\boldsymbol{A} = \begin{pmatrix} -y_1 \boldsymbol{x}_1^{\mathrm{T}} & y_1 \boldsymbol{x}_1^{\mathrm{T}} & -y_1 \\ \vdots & \vdots & \vdots \\ -y_n \boldsymbol{x}_n^{\mathrm{T}} & y_n \boldsymbol{x}_n^{\mathrm{T}} & -y_n \end{pmatrix} \in \mathbb{R}^{n \times 2d+1} \qquad \text{(inequality constraints)}$$

$$\boldsymbol{b} = \mathbf{0} \in \mathbb{R}^n$$

$$\boldsymbol{A}^{\mathrm{eq}} = \begin{pmatrix} \frac{1}{n} \sum_{i=1}^{n} y_i \boldsymbol{x}_i^{\mathrm{T}} & -\frac{1}{n} \sum_{i=1}^{n} y_i \boldsymbol{x}_i^{\mathrm{T}} & \frac{1}{n} \sum_{i=1}^{n} y_i \end{pmatrix} \in \mathbb{R}^{1 \times 2d+1} \qquad \text{(equality constraint)}$$

$$\boldsymbol{b}^{\mathrm{eq}} = 1$$

$$\boldsymbol{l} = \begin{pmatrix} \mathbf{0}^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -\infty \end{pmatrix} \qquad \text{(lower bounds of the variables)}$$

$$\boldsymbol{u} = \begin{pmatrix} \infty \cdots \infty & \infty \cdots \infty & \infty \end{pmatrix} \qquad \text{(upper bounds of the variables)}$$

Here, two slightly overlapping variable naming schemes are mixed to avoid uncommon notations. Thus, $\boldsymbol{x}$, the variable to be optimised, is not to be confused with the input data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and the bias b is to be distinguished from the equality constraint vector $\boldsymbol{b}$. Additionally, we use ∞ and $-\infty$ to indicate that the variables have no upper or lower bound, respectively. The above formulation is memory inefficient as it requires the inequality constraint matrix to be stored twice — once with a positive and once with a negative sign. The second minor issue is related to the number of non-zero entries in the constraint matrices. In general, linear programming solvers are more efficient on sparse matrices. An alternative formulation involving the training data only once uses the substitution $s_i = w_i^+ + w_i^-$. Thus, we get

$$w_i^+ = s_i - w_i^- \quad \Rightarrow \quad w_i = w_i^+ - w_i^- = s_i - 2w_i^- \quad \Rightarrow \quad \frac{1}{2}\left(s_i - w_i\right) = w_i^- \geq 0$$

and, vice versa,

$$w_i^- = s_i - w_i^+ \quad \Rightarrow \quad w_i = w_i^+ - w_i^- = 2w_i^+ - s_i \quad \Rightarrow \quad \frac{1}{2}\left(s_i + w_i\right) = w_i^+ \geq 0\,.$$

The transformed optimisation problem

$$\begin{aligned} &\text{minimise} && \sum_{i=1}^{d} s_i \\ &\text{subject to} && \begin{cases} y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geq 0 \quad \text{for all } i \\ \frac{1}{n}\sum_{i=1}^{n} y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) = 1 \\ s_i + w_i \geq 0 \quad \text{for all } i \\ s_i - w_i \geq 0 \quad \text{for all } i \end{cases} \end{aligned}$$

has the same optimum but is memory efficient and much sparser. The input matrices and vectors take the following values:

$$\boldsymbol{f} = \begin{pmatrix} \mathbf{0}^{\mathrm{T}} & \mathbf{1}^{\mathrm{T}} & 0 \end{pmatrix}^{\mathrm{T}} \in \mathbb{R}^{2d+1} \qquad \text{(objective function)}$$

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{w}^{\mathrm{T}} & \boldsymbol{s}^{\mathrm{T}} & b \end{pmatrix} \qquad \text{(target variable)}$$

$$\boldsymbol{A} = \begin{pmatrix} -y_1\boldsymbol{x}_1^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -y_1 \\ \vdots & \vdots & \vdots \\ -y_n\boldsymbol{x}_n^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & -y_n \\ \boldsymbol{I}_d & -\boldsymbol{I}_d & \mathbf{0} \\ -\boldsymbol{I}_d & -\boldsymbol{I}_d & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(n+2d)\times(2d+1)} \qquad \text{(inequality constraints)}$$

$$\boldsymbol{b} = \mathbf{0} \in \mathbb{R}^n$$

$$\boldsymbol{A}^{\text{eq}} = \begin{pmatrix} \frac{1}{n}\sum_{i=1}^{n} y_i \boldsymbol{x}_i^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & \frac{1}{n}\sum_{i=1}^{n} y_i \end{pmatrix} \in \mathbb{R}^{1\times 2d+1} \qquad \text{(equality constraint)}$$

$$\boldsymbol{b}^{\text{eq}} = 1$$

$$\boldsymbol{l} = \begin{pmatrix} -\infty \cdots -\infty & \mathbf{0}^{\mathrm{T}} & -\infty \end{pmatrix} \qquad \text{(lower bounds of the variables)}$$

$$\boldsymbol{u} = \begin{pmatrix} \infty \cdots \infty & \infty \cdots \infty & \infty \end{pmatrix} \qquad \text{(upper bounds of the variables)}$$

In the first approach, the constraint matrix $\boldsymbol{A}$ has $n(2d+1)$ non-zero entries while in the reformulated version $n(d+1)+4d$ entries are non-zero. Thus, for $n > 4$ the reformulated version has less entries. However, the complexity of linear programming solvers makes an a priori runtime prediction impossible due to numerous processing steps — presolving, scaling, solving. So, other less obvious aspects than the number of non-zero entries might get important in practise and require an empirical runtime evaluation (see Chapter 4.3).

The soft SFM approach is reformulated in the same way. In the initial problem

$$\begin{aligned} &\text{minimise} && \|\boldsymbol{w}\|_1 + C\|\boldsymbol{\xi}\|_1 \\ &\text{subject to} && \begin{cases} y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geq -\xi_i \quad \text{for all } i \\ \frac{1}{n}\sum_{i=1}^{n} y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) = \pm 1 \\ \xi_i \geq 0 \quad \text{for all } i \end{cases} \end{aligned}$$

the substitution $w_i = w_i^+ - w_i^-$ with $w_i^+, w_i^- \geq 0$ leads to

$$\begin{aligned} &\text{minimise} && \sum_{i=1}^{d}\left(w_i^+ + w_i^-\right) + C\sum_{i=1}^{n}\xi_i \\ &\text{subject to} && \begin{cases} y_i\left(\sum_{j=1}^{d}\left(w_j^+ - w_j^-\right)x_{ij} + b\right) \geq -\xi_i \quad \text{for all } i \\ \frac{1}{n}\sum_{i=1}^{n} y_i\left(\sum_{j=1}^{d}\left(w_j^+ - w_j^-\right)x_{ij} + b\right) = \pm 1 \\ w_i^+, w_i^-, \xi_i \geq 0 \quad \text{for all } i \quad . \end{cases} \end{aligned}$$

Again, the size and the structure of the inequality constraint matrix $\boldsymbol{A}$ is the crucial factor:

$$\boldsymbol{A} = \begin{pmatrix} -y_1\boldsymbol{x}_1^{\mathrm{T}} & y_1\boldsymbol{x}_1^{\mathrm{T}} & & -y_1 \\ \vdots & \vdots & -\boldsymbol{I}_n & \vdots \\ -y_n\boldsymbol{x}_n^{\mathrm{T}} & y_n\boldsymbol{x}_n^{\mathrm{T}} & & -y_n \end{pmatrix} \in \mathbb{R}^{n\times 2d+n+1} \tag{3.17}$$

Here, $n(2d+2)$ entries are non-zero, and, as in the hard-margin case, the input data needs to be stored twice. This is again avoided by substituting $s_i = w_i^+ + w_i^-$ to get the linear program

$$\begin{aligned} &\text{minimise} && \sum_{i=1}^{d} s_i + C \sum_{i=1}^{n} \xi_i \\ &\text{subject to} && \begin{cases} y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) \geq -\xi_i \quad \text{for all } i \\ \frac{1}{n} \sum_{i=1}^{n} y_i \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \right) = \pm 1 \\ s_i + w_i \geq 0 \quad \text{for all } i \\ s_i - w_i \geq 0 \quad \text{for all } i \\ \xi_i \geq 0 \quad \text{for all } i. \end{cases} \end{aligned}$$

Now, the inequality constraint matrix is

$$\boldsymbol{A} = \begin{pmatrix} -y_1 \boldsymbol{x}_1^{\mathrm{T}} & & & -y_1 \\ \vdots & \boldsymbol{0}_{n,d} & -\boldsymbol{I}_n & \vdots \\ -y_n \boldsymbol{x}_n^{\mathrm{T}} & & & -y_n \\ -\boldsymbol{I}_d & -\boldsymbol{I}_d & \boldsymbol{0}_{d,n} & \boldsymbol{0} \\ \boldsymbol{I}_d & -\boldsymbol{I}_d & \boldsymbol{0}_{d,n} & \boldsymbol{0} \end{pmatrix} \in \mathbb{R}^{(n+2d)\times(2d+n+1)} \tag{3.18}$$

with $4d + dn + 2n$ non-zero entries.

Technical Issues As with many machine learning algorithms, normalisation is an essential preprocessing step for any of the proposed SFM variants. For all experiments, we normalised the training datasets to zero mean and unit variance and finally scaled all vectors to have a mean norm of one. This last step is sometimes beneficial in high-dimensional scenarios to keep the outcome of scalar products in a reasonable range. The test sets were normalised according to the factors obtained from the corresponding training sets.

In theory, for hard SFMs, either no solution exists or a solution where all data points are correctly classified. However, since the optimiser uses numerical approximation methods with certain accuracy thresholds, some constraints may be marginally violated. Thus, some data points may be located on the wrong side of the hyperplane, but very close to it, producing a non-zero training error even in the hard case.

To avoid numerical issues, numbers that differed by no more than a specific implementation-dependent number, normally closely connected to the machine epsilon ε, were considered to be equal.

3.6 Conclusions

The SFM aims to classify given input data with the least number of features. It approximates the zero-norm minimising weight vector of a separating hyperplane in the same way as the approach by WESTON et al. — by minimising its one-norm. But, in contrast to support vector based feature selection, it does not maximise the margin while simultaneously selecting features. We regard this as the major advantage of the SFM — to focus exclusively on feature selection. The SFM implements the principle of structural risk minimisation by deriving a nested hypothesis space. Each iteration requires solving a linear program — solvers for such problems are available in many flavours both commercially and free. The advantages of the SFM can be made plausible by considering the coincidence of zero-norm and one-norm minimising solution in some specific scenarios. Although no universal proof could be given, we expect the SFM to find the optimal solution more likely than WESTON's method for a wide rage of scenarios. However, feature selection has theoretical limits — the VC-dimension of combined classification and feature selection is larger than if the irrelevant features would not have been included in the training at all. Thus, we cannot expect any feature selection method to provide the same generalisation performance as if no irrelevant features were present at all.

A man of genius makes no mistakes. His errors are volitional and are the portals of discovery.

«Ulysses», James Joyce

4 Basic Experiments

We introduced the support feature machine as a novel feature selection method, showed why a superior performance with respect to the SVM-based method by Weston et al. is theoretically to be expected and demonstrated how this machine can be engineered using linear programming solvers. What now follows is an in depth analysis of the performance of the support feature machine on various artificial and real world datasets. However, as most of them are high-dimensional small sample size scenarios, we start with experiments that stress the unintuitive behaviour of such datasets. In particular, we give practical bounds to decide when leave-one-out cross-validation for support vector machines may completely fail. Datasets having higher dimensionality will behave as if they were infinite dimensional.

After these introductory experiments, we compare the performance of the basic SFM with Weston's SVM-based feature selection method on artificial data as both methods are very closely related. Additionally, we compare both methods to a standard linear support vector machine to demonstrate the necessity to reduce the number of features even in simple scenarios. We show how the three methods perform as the number of irrelevant features grows exponentially. The basic experimental analysis concludes with an evaluation of the soft SFM.

Basically, a support feature machine is a linear program to be solved with any commercial or freely available linear programming solver. However, their actual runtime is hard to predict in advance. Therefore, we benchmark the runtime depending on the dataset size, the dimensionality and an a priori known number of truly relevant features. We compare four popular linear programming solvers and give advice which to use in particular scenarios.

After having evaluated the support feature machine on artificial datasets, we finally apply it to a publicly available microarray dataset.

4.1 Reliability of Cross-Validation

In high-dimensional small sample size scenarios distances between data points tend to concentrate — they all become nearly the same. In the limit, for an infinite number of irrelevant dimensions, i.e. when $d \to \infty$, they become exactly the same (see Chapter 2.5.2). No real-world dataset is actually infinite dimensional, but the point from which on a finite dimensional dataset practically behaves as if it was infinite dimensional is of particular interest.

First, we analyse how leave-one-out cross-validation performs on pure random data. Therefore, we sampled two classes independently and identically distributed from the standard normal distribution and trained support vector machines for $n \in \{4, 6, \ldots, 40\}$ and logarithmically spaced $d \in [10, 10^5]$. For each configuration the procedure was repeated 100 times and the average leave-one-out error was determined. The colour-coded results are shown in Figure 4.1. For any fixed sample size (e.g. $n = 20$) the leave-one-out error reaches 1 as soon as a specific

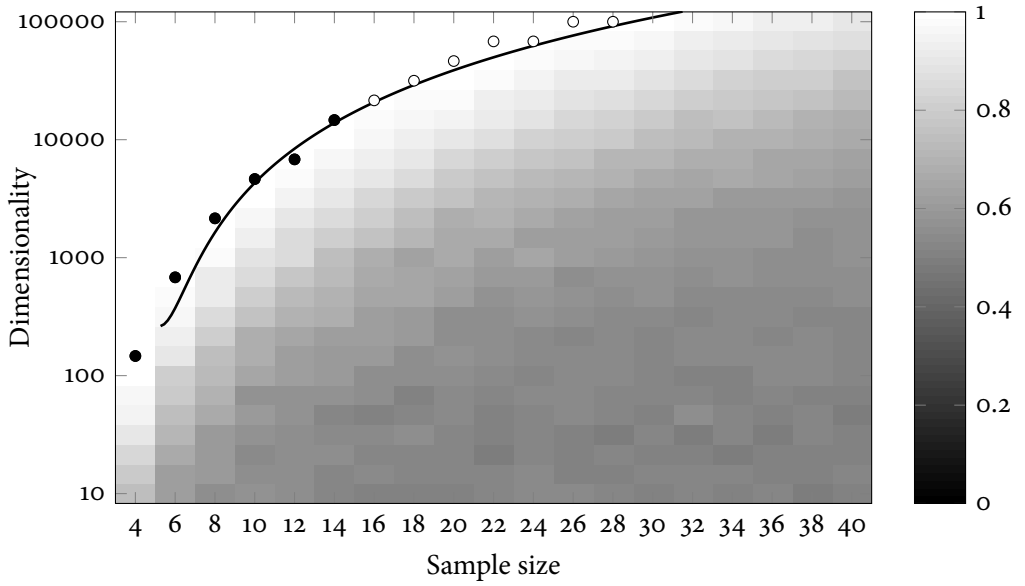


Figure 4.1: Average leave-one-out cross-validation error depending on sample size and dimensionality for random data. White indicates an error rate of 1, i.e. the corresponding parameter sets behave as if they had infinite dimensionality. Additionally, the border between normal and infinite-like behaviour is approximated by a second order polynomial on those scenarios where the mean error rate exceeds 0.99 for the first time. This model — derived from the first 6 sample sizes (black dots) — fits well even in the extrapolation case (white dots).

dimensionality is exceeded (in this case for approximately $d = 50000$). A precise border between regular and infinite-like behaviour does not exist, however, we can approximate the border with a simple least-squares fit. For each sample size we picked the lowest dimensionality such that the corresponding average error exceeds 0.99. We derived a least squares fit using the first 6 points and the 2nd order polynomial $d_{\text{poly}}(n) = \alpha_0 + \alpha_1 \cdot n + \alpha_2 \cdot n^2$. This heuristic suits for illustrating the asymptotic behaviour but should not be taken as an exact mathematical coherence.

In the second experiment, we consider more realistic scenarios where the two classes have different means. In this case, according to Theorem 2.5.2, the error rate converges to 1 for small sample sizes and — as soon as the sample size exceeds a certain threshold — to zero as dimensionality goes to infinity. Again, it remains to show how large the dimensionality must get to cause infinite-like behaviour. Given two equally sized classes each drawn from the multivariate standard normal distribution ($\sigma^2 = \tau^2 = 1$) with an inter-class distance such that $\mu^2 = \frac{1}{30}$ (see Figure 4.2, again with samples $n \in \{4, 6, \ldots, 40\}$ and logarithmic-spaced $d \in [10, 10^5]$ for 100 independent runs). Obviously, the average error rate of all scenarios with a sample size of at most 12 converges to 1, while in all other scenarios the error rate converges to 0. The same threshold can be derived using the above class distribution parameters and the fact, that in leave-one-out cross-validation one class contains $\frac{n}{2}$ data points while the other contains one element less. Thus, according to Theorem 2.5.2, the critical sample size derives as

$$\frac{\sigma^2}{k} - \frac{\tau^2}{l} = \mu^2 \quad \Rightarrow \quad \frac{1}{\frac{n}{2} - 1} - \frac{1}{\frac{n}{2}} = \frac{1}{30} \quad \Rightarrow \quad n = 12\,.$$

Therefore, the probability of a new data point to be classified correctly by an SVM using leave-one-out cross-validation converges to 1 for $n > 12$ and to zero for $n \leq 12$. Figure 4.2 visualises the critical scenarios where infinite-like behaviour is to be expected (black and white regions).

Finally, we evaluated the behaviour of a soft-margin SVM. On random data, a softness value of C causes a d-dimensional scenario to behave as if it was $\left(d + \frac{1}{C}\right)$-dimensional (see Chapter 2.5.5). In real two class scenarios, the behaviour was very sensitive to the actual choice of C (see Figure 4.3), i.e. slightly changing C causes a dramatic change of the convergence behaviour towards 1 and 0, respectively. In many SVM applications, the softness parameter is varied on a logarithmic grid from small (e.g. 10^{-8}, allowing many outliers) to large values (e.g. 10^8, allowing almost no outlier) to find the optimum. Due to runtime considerations the grid spacing cannot be chosen arbitrary small. In general, a coarse grid is initially used which is later refined around optimum candidates. This technique assumes the performance to change only gradually between neighbouring points. However, our experiment shows that for leave-one out cross-validation in high-dimensional small sample size scenarios this is not the case — the

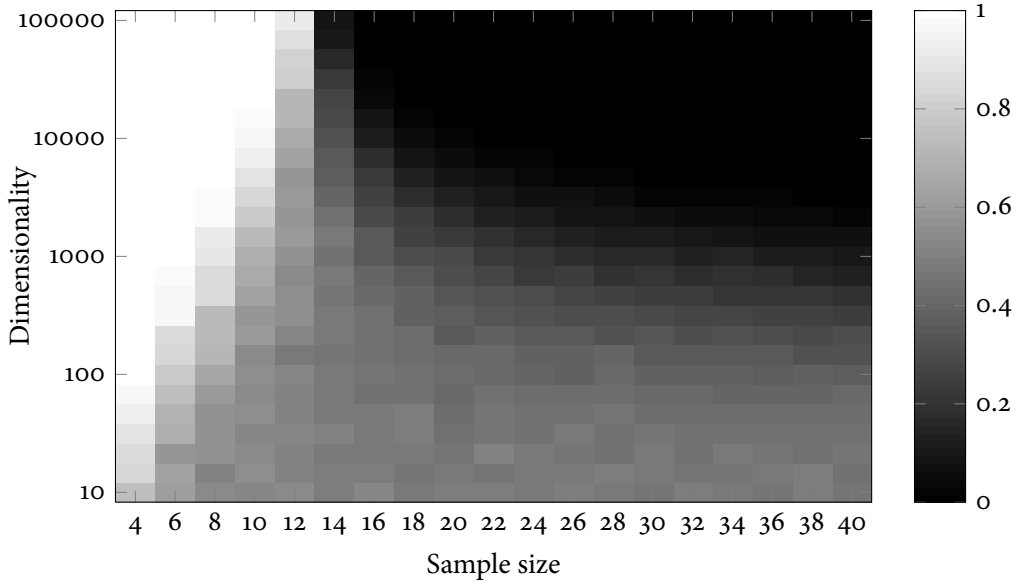


Figure 4.2: Average leave-one-out cross-validation error depending on sample size and dimensionality for real two class scenarios. Both classes have a distance corresponding to $\mu^2 = \frac{1}{30}$. White indicates an error rate of 1, black an error rate of 0. Parameter sets corresponding to black or white behave as if they had infinite dimensionality, while medium grey indicates the regular low-dimensional behaviour (averaged over 100 runs).

convergence behaviour whether we choose $C = 1$ or $C = 0.1$ is completely different. So, not only the validation procedure fails but also the higher order methods such as grid search may fail.

4.2 Support Feature Machine on Artificial Data

Having introduced the basic principles of the SFM and its theoretical advantages (see Chapter 3), we now show that the SFM outperforms both the standard SVM and WESTON's SVM-based feature selection method in many simulations, particularly in high-dimensional small sample size scenarios. We first compare the capability of the SFM and WESTON's method to identify the truly relevant features in a very simple scenario with linearly separable, equally sized classes. We then show how the three methods perform if an exponentially increasing number of irrelevant features is added to the data, which is probably the most challenging setting for any feature selection method. The experimental analysis on artificial data concludes with an evaluation of the soft SFM.

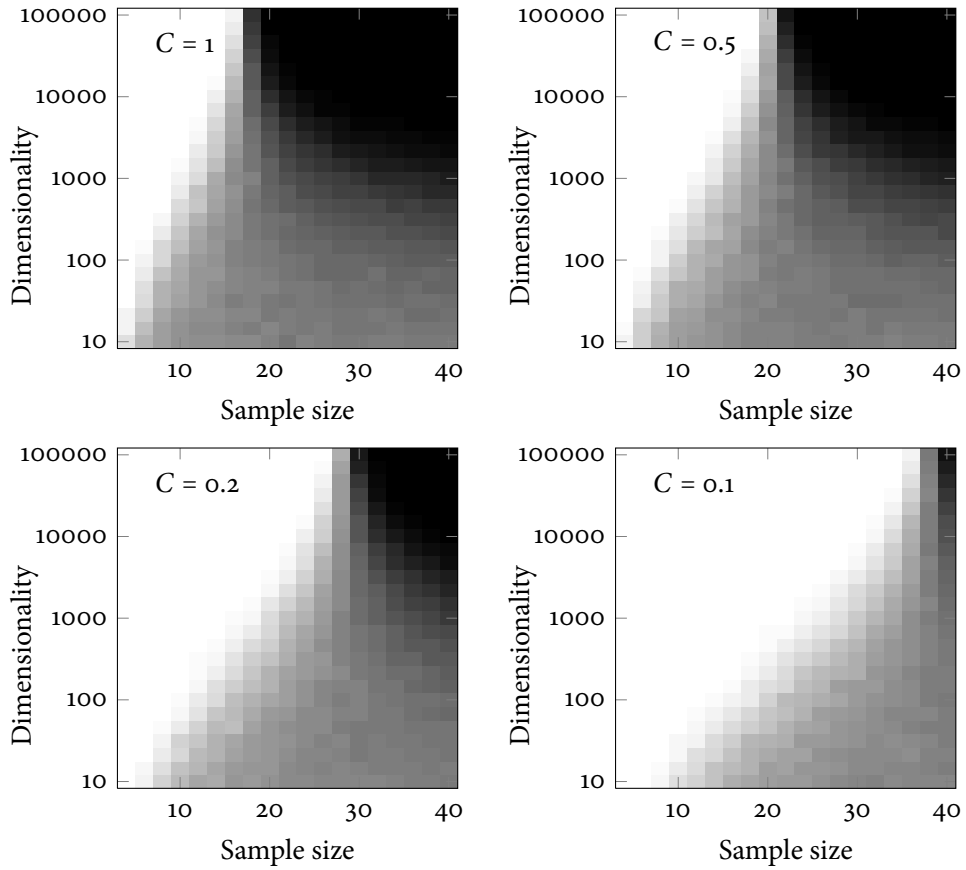


Figure 4.3: Average leave-one-out cross-validation error depending on sample size, dimensionality and softness for real two class scenarios. Both classes have a distance corresponding to $\mu^2 = \frac{1}{30}$. Shown are the results for slightly varying softness parameters (C = 1.0, 0.5, 0.2, 0.1, averaged over 100 runs).

4.2.1 Basic Experiment

For the basic experiment, we generated artificial datasets with balanced classes. The first k dimensions $x_i, \ldots, x_k$ were drawn as $x_i = \mathcal{N}(\mu \cdot y, 1)$. The remaining features $x_{k+1}, \ldots, x_d$ were noise drawn as $x_i = \mathcal{N}(0, 1)$. The parameter μ determines the distance between the means of the two classes. We ensured that both classes were linearly separable within the first k dimensions. However, both classes might be separable with even less than k features by chance. The number of dimensions was set to $d = 1000$. In the first experiment, we used a fixed sample size of $n = 100$ and varied the number of relevant features from $k = 1$ to $k = 20$. In the second experiment, we set the number of relevant features to $k = 5$ and varied the sample size from $n = 10$ to $n = 1000$.

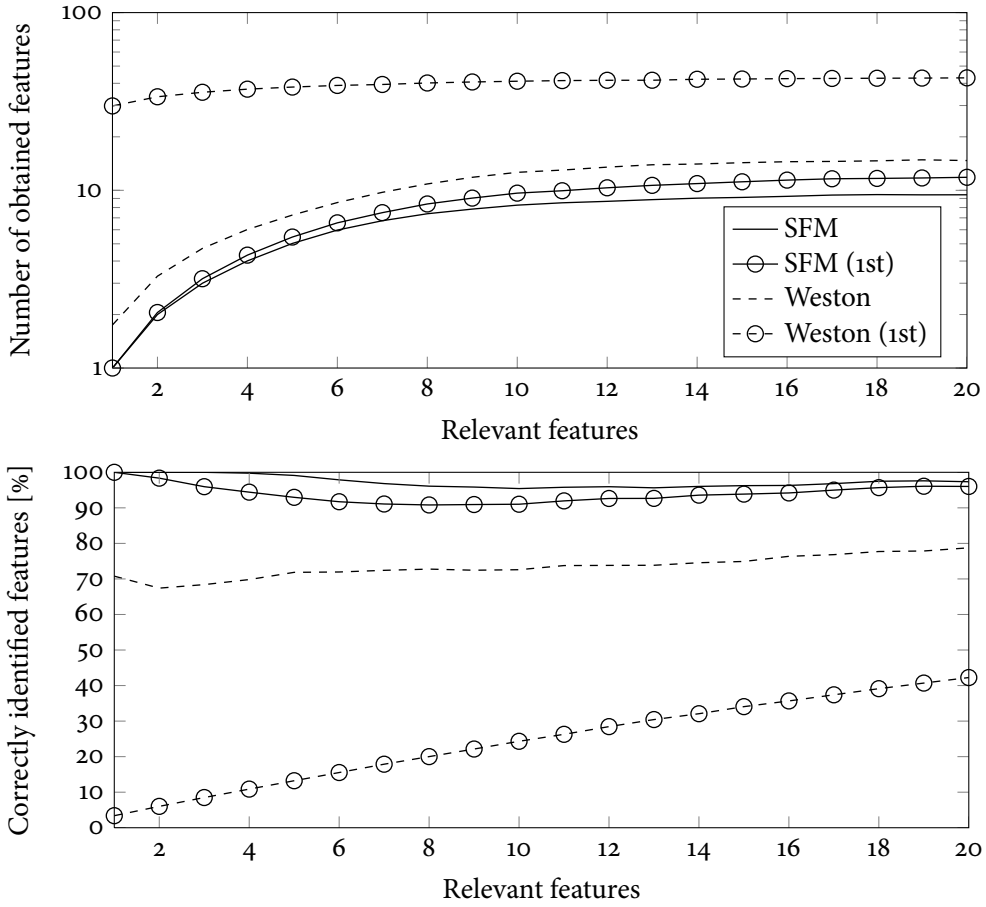


Figure 4.4: Feature selection performance of the SFM and WESTON's method depending on the number of truly relevant features. Shown are the average number (1000 runs) of obtained features and the average percentage of correctly identified features as percentage of obtained features for both methods after the first and after the final iteration ($n = 100$, $d = 100$, $\mu = 0.3$).

Testing the basic SFM and WESTON's method with this data revealed that the ability to identify relevant features differed between both methods in many aspects (see Figures 4.4 and 4.5). Obviously, the SFM returned both a smaller total number of features and a higher percentage of correctly identified features for almost all scenarios. WESTON's method returned more irrelevant features than the SFM in almost all individual scenarios. In scenarios with very low intrinsic dimensionality — for $k = 1, \ldots, 4$ — the SFM identified all features correctly (see Figure 4.4, bottom).

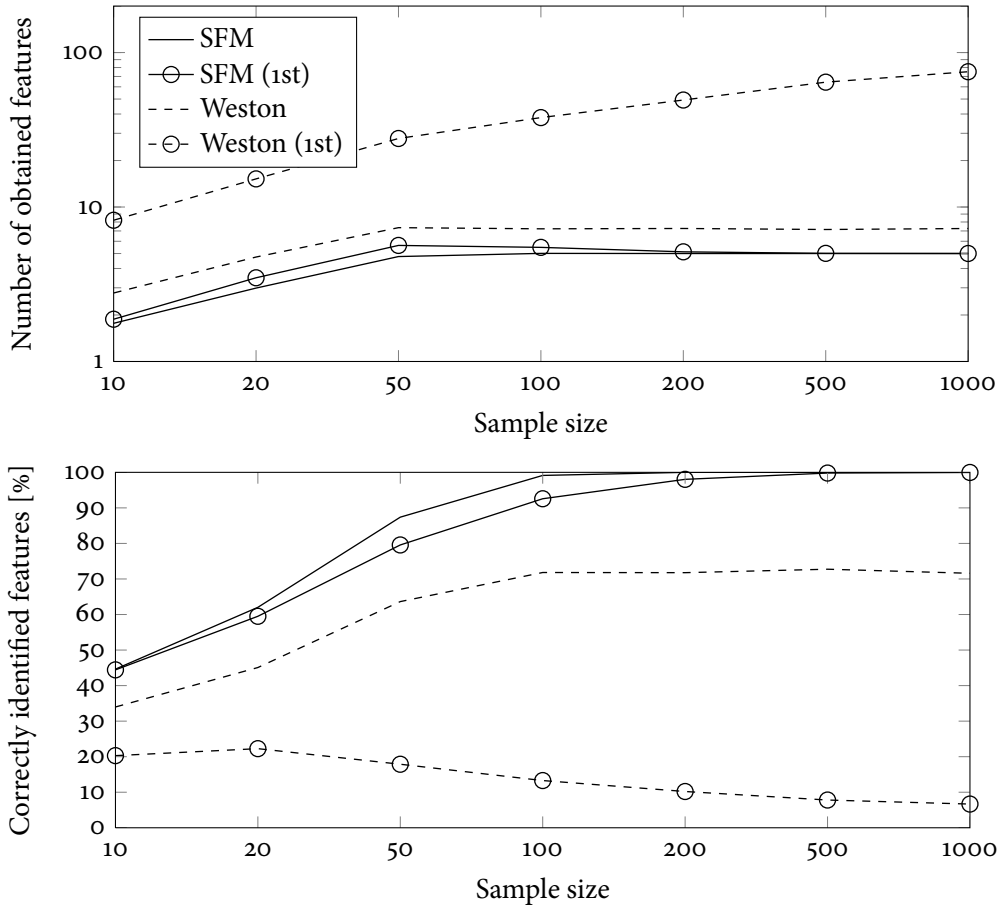


Figure 4.5: Feature selection performance of the SFM and WESTON's method depending on the sample size. Again, the average number (1000 runs) of features and the average percentage of correctly identified features are shown for both methods after the first and after the final iteration ($d = 100$, $k = 5$, $\mu = 0.3$).

In scenarios with $k = 5$, the SFM identified all relevant features correctly in each and every run, if the number of data points exceeded 200 (see Figure 4.5). WESTON's method failed to converge to the correct number of features even if the number of data points was further increased (to 1000) — the percentage of correctly identified features stayed clearly below 80%. Thus, the SFM converged to the correct set of features, while the SVM-based approach got stuck in a local minimum even for large datasets. In contrast to WESTON's method, the SFM was close to the final solution already in the very first iteration. In scenarios with a large number of data points, the SFM converged already after one iteration (see Figure 4.5, $n = 1000$).

4.2.2 Increasing the Dimensionality

For many biomedical applications, a feature selection method should not only be able to deal with high-dimensional datasets but should also *scale well*, i.e. adding irrelevant features should not significantly degrade the performance. To assess how the performance of the SVM, WESTON's method, and the SFM degrade when irrelevant features are added to the data, we applied an artificial dataset that has been used in a variety of publications but was originally proposed in [WESTON et al., 2003].

The dataset contains two equally-sized classes, where the first six dimensions are relevant and the remaining dimensions contain Gaussian noise. With probability 0.7, the first three features are drawn as $x_i = y\mathcal{N}(i,1)$ and the second three features are drawn as $x_i = \mathcal{N}(0,1)$. With probability 0.3 the setting is inverted, i.e. the first three features are drawn as $x_i = \mathcal{N}(0,1)$ and the second three as $x_i = y\mathcal{N}(i-3,1)$. The remaining k^* features are drawn as $x_i = \mathcal{N}(0,20)$ with $k^* = 10, 10^2, 10^3, 10^4$. Additionally, we ensured the training set to be linearly separable within the six informative dimensions. We sampled n training points ($n = 20, 50, 100, 200, 500$) and 5000 test data points. Three classifiers (a standard SVM without feature selection, WESTON's method, and the basic SFM) were trained on each dataset.

Figure 4.6 compares the capability of both feature selection methods to identify relevant features. Shown are the average results of 100 runs. Compared to WESTON's approach, the SFM returns (i) a smaller number of features and (ii) more likely the truly relevant ones. Even in very high-dimensional small sample size scenarios the SFM can identify the relevant dimensions very effectively: As the number of data points increases, the number of features found to be relevant increases, but does not exceed 6 — the number of truly relevant features. The percentage of correctly identified features decreases when the number of noise dimensions increases. However, only in extremely high-dimensional small sample size scenarios the percentage of correctly identified features drops below 90%. Note, the sampling scheme causes the number of correctly identified features not to converge to 100% for large n. Due to the experimental design some features provide a better separability than others. Then, by chance, one of the irrelevant features may be favoured by the SFM (and also WESTON's method) instead of the weakest relevant feature. This effect gets amplified for large sample sizes as the solution space becomes smaller and smaller.

Figure 4.7 shows the impact of irrelevant features on the test error. Obviously, the SVM performs poorly for $k^* \geq 1000$. Both the SFM and WESTON's method perform significantly better in these scenarios: They show a significantly lower test error than the SVM, and test errors increase significantly more slowly with the number of irrelevant features than for the SVM. For $n \geq 50$ the test error of the SFM and WESTON's method show no increase with increasing number of irrelevant features, even if k^* is increased from 10 to 10000.

(a) SFM, number of obtained features

n	$k^* = 10$	$k^* = 100$	$k^* = 1000$	$k^* = 10000$
20	2.0 (± 0.6)	2.0 (± 0.6)	2.0 (± 0.6)	1.9 (± 0.6)
50	2.3 (± 0.6)	2.4 (± 0.6)	2.4 (± 0.6)	2.5 (± 0.7)
100	2.7 (± 0.7)	2.6 (± 0.6)	2.6 (± 0.7)	2.6 (± 0.7)
200	3.1 (± 0.7)	3.2 (± 0.8)	3.2 (± 0.8)	3.1 (± 0.7)
500	4.1 (± 0.8)	4.2 (± 0.8)	4.2 (± 0.8)	4.2 (± 0.8)

(b) WESTON, number of obtained features

n	$k^* = 10$	$k^* = 100$	$k^* = 1000$	$k^* = 10000$
20	2.6 (± 0.8)	2.8 (± 0.9)	3.0 (± 1.1)	3.2 (± 1.1)
50	3.2 (± 1.0)	3.4 (± 1.1)	3.3 (± 1.1)	3.4 (± 1.1)
100	3.8 (± 1.1)	4.0 (± 1.2)	4.0 (± 1.2)	3.9 (± 1.2)
200	4.6 (± 1.2)	4.9 (± 1.4)	4.9 (± 1.4)	4.8 (± 1.2)
500	5.9 (± 1.3)	6.2 (± 1.5)	6.4 (± 1.5)	6.2 (± 1.4)

(c) SFM, percentage of correctly identified relevant features

n	$k^* = 10$	$k^* = 100$	$k^* = 1000$	$k^* = 10000$
20	98.5% (± 8.0%)	88.9% (±20.2%)	66.5% (±29.1%)	46.9% (±32.9%)
50	99.6% (± 3.5%)	98.8% (± 6.5%)	96.7% (±10.9%)	85.9% (±22.4%)
100	99.7% (± 3.1%)	99.1% (± 5.3%)	97.3% (± 8.8%)	96.9% (± 9.4%)
200	99.4% (± 3.9%)	98.5% (± 6.0%)	96.6% (± 8.9%)	95.1% (±11.0%)
500	98.8% (± 5.0%)	96.4% (± 8.7%)	94.3% (±10.8%)	92.2% (±11.7%)

(d) WESTON, percentage of correctly identified relevant features

n	$k^* = 10$	$k^* = 100$	$k^* = 1000$	$k^* = 10000$
20	94.8% (±12.8%)	81.2% (±23.7%)	58.0% (±29.6%)	32.5% (±26.1%)
50	94.0% (±12.1%)	87.6% (±16.0%)	85.0% (±18.0%)	79.7% (±20.3%)
100	93.1% (±12.0%)	87.0% (±16.0%)	83.4% (±17.5%)	83.1% (±17.1%)
200	89.0% (±13.6%)	82.9% (±15.9%)	80.3% (±16.0%)	80.6% (±16.0%)
500	82.2% (±12.9%)	76.2% (±14.5%)	73.5% (±14.7%)	74.1% (±14.6%)

Figure 4.6: Impact of an exponentially increasing number of irrelevant features on the variable selection performance. Shown are the average number of features (1000 runs) found to be relevant (± std) by a hard SFM (a) and by WESTON's method (b) and the average percentage of correctly identified features (± std) for a hard SFM (c) and for WESTON's method (d).

(a) Support vector machine

n	$k^* = 10$	$k^* = 100$	$k^* = 1000$	$k^* = 10000$
20	4.1% (± 2.1%)	23.3% (± 2.7%)	40.6% (± 1.1%)	47.0% (± 0.7%)
50	1.5% (± 0.6%)	12.4% (± 2.0%)	34.5% (± 1.1%)	44.9% (± 0.8%)
100	0.9% (± 0.3%)	6.5% (± 1.2%)	28.5% (± 1.1%)	42.8% (± 0.8%)
200	0.7% (± 0.2%)	3.2% (± 0.6%)	21.4% (± 1.0%)	39.7% (± 0.7%)
500	0.5% (± 0.1%)	1.5% (± 0.3%)	11.9% (± 0.8%)	34.0% (± 0.7%)

(b) Weston's method

n	$k^* = 10$	$k^* = 100$	$k^* = 1000$	$k^* = 10000$
20	5.0% (± 4.3%)	8.4% (± 7.6%)	17.5% (±12.1%)	30.6% (±13.7%)
50	2.2% (± 1.5%)	2.5% (± 1.6%)	2.7% (± 1.8%)	3.5% (± 3.3%)
100	1.5% (± 0.7%)	1.5% (± 0.7%)	1.6% (± 0.9%)	1.7% (± 0.9%)
200	1.1% (± 0.4%)	1.1% (± 0.5%)	1.1% (± 0.5%)	1.2% (± 0.5%)
500	0.8% (± 0.3%)	0.8% (± 0.3%)	0.8% (± 0.3%)	0.8% (± 0.3%)

(c) Support feature machine

n	$k^* = 10$	$k^* = 100$	$k^* = 1000$	$k^* = 10000$
20	12.4% (± 5.4%)	14.2% (± 6.2%)	19.0% (± 9.9%)	27.7% (±15.3%)
50	5.3% (± 2.9%)	5.5% (± 3.3%)	6.1% (± 3.6%)	8.4% (± 5.2%)
100	3.0% (± 1.6%)	2.9% (± 1.6%)	2.9% (± 1.5%)	3.0% (± 1.9%)
200	1.7% (± 0.7%)	1.6% (± 0.7%)	1.7% (± 0.8%)	1.7% (± 0.8%)
500	1.0% (± 0.3%)	1.0% (± 0.3%)	1.0% (± 0.4%)	1.0% (± 0.3%)

Figure 4.7: Impact of an exponentially increasing number of irrelevant features on the test error. Shown are the test error (± std) of an SVM without feature selection (a), of the method proposed by Weston (b), and of the hard SFM (c) on an artificial dataset with 6 relevant k^* irrelevant features averaged over 1000 runs.

4.2.3 Non-separable Classes

Finally, we constructed an artificial problem where the two classes were not linearly separable. The probabilities of the classes $y = 1$ and $y = -1$ were equal, both in the training and the test set. The first k dimensions $x_1, \ldots, x_k$ were drawn normally distributed as $x_i = \mathcal{N}(\mu \cdot y, 1)$. The remaining features $x_{k+1}, \ldots, x_d$ were noise drawn as $x_i = \mathcal{N}(0, 1)$. The parameter μ was used to adjust the distance between the class centres. Both, the training and the test sets were sampled according to the above procedure, each containing n data points. The softness parameter C was sampled in 100 steps logarithmically spaced between 0.01 and 100.

We tested four scenarios. In the first scenario, the data contains a high percentage of relevant features. In the second scenario, we reduced the number of relevant features and increased the number of irrelevant features. Finally, in the third and fourth scenario we further increased the number of irrelevant features and reduced the number of data points.

Figure 4.8 shows the averaged results of 100 runs with a soft SFM. In all four scenarios, increasing softness — allowing more training errors — results in a smaller number of obtained features. This is in line with the theoretical consideration that the soft SFM directly trades off the number of features and the training error, and that a very soft SFM will return a single feature. As the number of obtained features decreases, the percentage of correctly identified features increases. However, this does not necessarily result in increased prediction performance: In the second scenario the test error reaches a minimum just after the point where the percentage of correctly identified features sharply increases. In the first and third scenario, however, the test error increases with increasing softness, and in the fourth scenario the test error remains almost constant throughout all softness values. Below, we discuss the four data scenarios in detail.

Many Relevant Features, Few Irrelevant Features, Large Sample Size In this case, the data contains a large percentage of relevant features and the number of data points exceeds the number of dimensions. However, features might be correlated and not all features might actually be required to separate the classes. The SFM identified almost no irrelevant feature as being relevant, independent of the softness. However, only a small fraction of the truly relevant features was identified (at most 7 out of 20). The test error decreased slightly with decreasing softness (see Figure 4.8, top left).

Few Relevant Features, Many Irrelevant Features, Large Sample Size In this scenario, the data contains few relevant and many irrelevant features but the number of data points still exceeds the number of dimensions (see Figure 4.8, top right). Increasing the softness resulted in a smaller set of obtained features, while the percentage of correctly identified features increased. For the parameters chosen here, the optimal test error is achieved for medium softness ($C = 0.2$), approximately at that point where almost all truly relevant features and almost no irrelevant features are obtained.

Few Relevant Features, Many Irrelevant Features, Small Sample Size The third and fourth scenario both represent high-dimensional small sample size data with few relevant and many irrelevant features (see Figure 4.8, bottom). This is a very challenging scenario because the information content is very small. As before, increasing softness resulted in a smaller number of obtained features while the percentage of correctly identified features increased. In the third

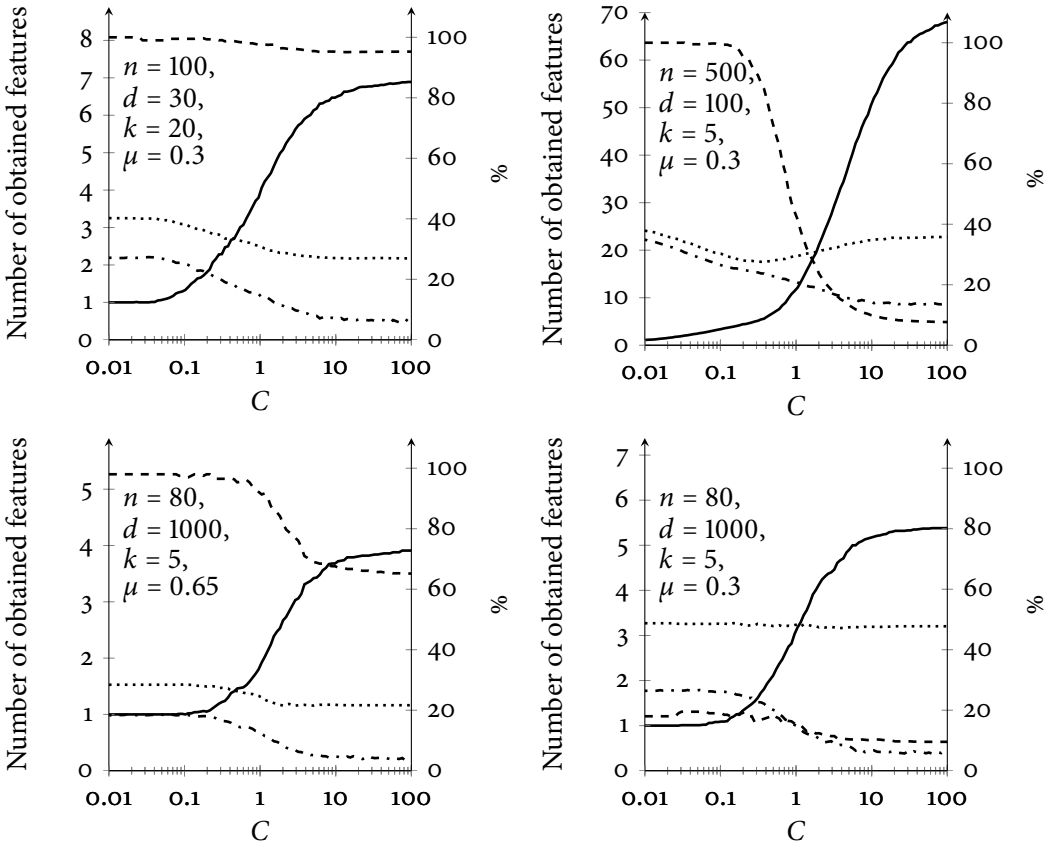


Figure 4.8: Behaviour of the soft SFM for linearly non-separable classes. Shown are four different scenarios and the resulting number of obtained features (solid line, left axis), percentage of correctly identified features (dashed, right axis), training error (dash-dotted, right axis), and test error (dotted, right axis) averaged over 100 runs.

scenario (see Figure 4.8, bottom left), we chose a larger class distance than in the previous scenarios (μ was increased from 0.30 to 0.65). In this scenario, the test error increased slightly with increasing softness, but stayed well below chance. In the fourth scenario we used the same class distance as in the first two scenarios. Although the percentage of correctly identified features still increased with increasing softness, the test error remained almost constant and never fell well below chance (see Figure 4.8, bottom right). Thus, in this scenario the class overlap seems to be too large to obtain meaningful results.

In sum, applying a soft SFM to the four exemplary datasets shows that the soft SFM performs well on datasets with non-separable classes even in scenarios with few relevant and many irrel-

evant features, as long the class overlap is not too large relative to the number of data points. In high-dimensional small sample size scenarios with few relevant features and strongly overlapping classes, the performance of the SFM drops to chance, possibly because the information content is no longer sufficient to allow valid feature selection. However, in such scenarios we expect all feature selection methods to be significantly affected by the low information content.

4.3 Runtime Simulations

The above experiments have shown various aspects of the support feature machine, especially its ability to identify relevant features and robustness against noise features. In practise, runtime and memory requirements may be equally important. A support feature machine may be implemented using linear programming solvers, however, the performances of conventional linear programming solvers may differ significantly and may also depend on the particular dataset configuration. For choosing the appropriate solver, we implemented the SFM using four linear programming toolboxes — CPLEX, MOSEK, MATLAB and GLPK (see Chapter 3.5). The following experiments show how the solvers perform with respect to sample size, dimensionality, intrinsic dimensionality, linear program formulation and class overlap.

The data was drawn in the same way as in the previous section with the additional constraint to be linearly separable. Thus, a d-dimensional balanced training set with n data points was sampled, where the first k dimensions $x_1, \ldots, x_k$ were drawn normally distributed as $x_i = \mathcal{N}(\mu \cdot y, 1)$ with $\mu \in \mathbb{R}^+$. The remaining features $x_{k+1}, \ldots, x_d$ were noise drawn as $x_i = \mathcal{N}(0, 1)$.

The performance of a linear programming solver does not exclusively depend on the dataset configuration but also on a variety of solver specific tuning parameters. Commercial solvers allow for tuning hundreds of parameters, including the choice of solver variants, presolvers, and numerous tolerance and termination thresholds. Changing the default values for a particular dataset might improve accuracy or runtime significantly. However, a systematic tuning of all parameters is in practise infeasible due to the large number of parameters. Additionally, parameter tuning is always biased according to the experience level of the user and will therefore bias every performance measure. In the following experiments no parameter tuning was applied — each solver was used with the default setting. Each solver was called within a MATLAB script. Only the time consumed by the entire solver was measured, not the overhead for data pre- and postprocessing. Further, we avoided swapping artefacts — i.e. runtime overheads caused by copying from the main memory to the hard drive — by limiting the problem size such that the problem fits into the main memory at any time during optimisation.

Note, this is not a universal benchmark for linear programming solvers. The runtime measurements are specific to the support feature machine. Other linear programs might cause a

completely different workload, and even the tendencies might get reversed. All experiments were run on an Intel Core 2 Quad machine with 2.4 GHz and 4 GB RAM running Linux Ubuntu 10.10. All runtime measurements were averaged over 100 runs.

Sample Size Increasing the sample size caused an almost linear increase in overall runtime (see Figure 4.9, upper curves) for all linear programming solvers and the standard linear program formulation. The absolute runtimes significantly differed among the four solvers. For a sample size of $n = 500$, MOSEK solved the problem in 84 ms on average, while the second best solver CPLEX took twice the time (176 ms). The remaining solvers — MATLAB and GLPK — took 4.9 s (≈ 60 ×) and 41.6 s (≈ 500 ×), respectively. This order stayed the same for a wide range

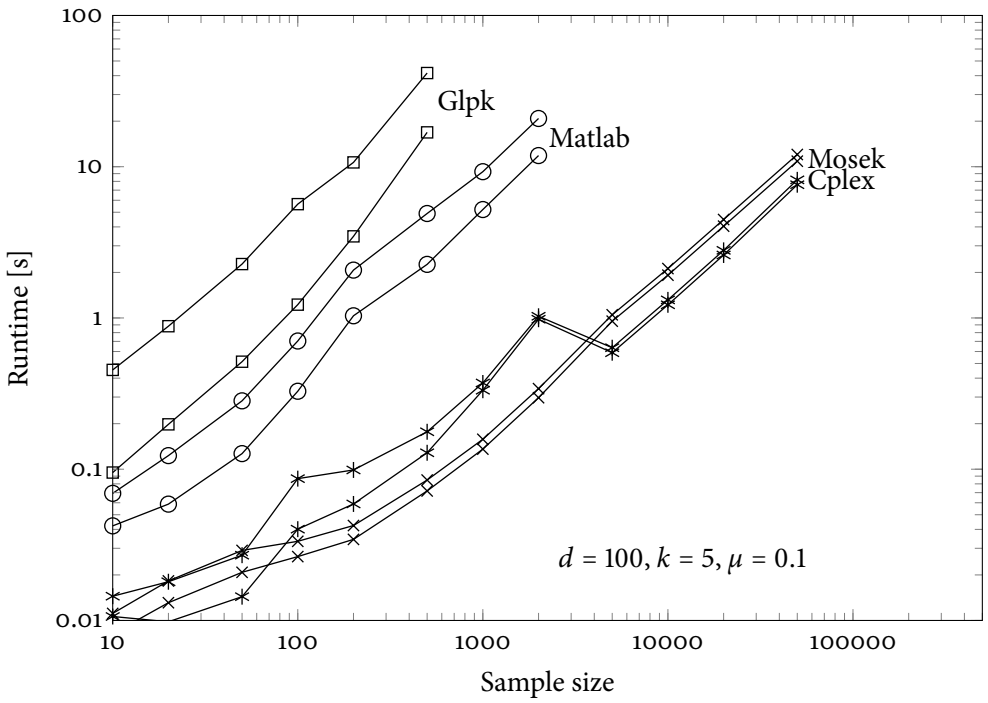


Figure 4.9: Average runtime of linear programming solvers depending on the sample size. The runtime increases approximately linearly with the sample size because the slopes in the above log-log graph are approximately 1 (slope in the last displayed interval: 1.08 (MOSEK), 1.17 (MATLAB), 1.17 (CPLEX), 1.48 (GLPK)). The absolute numbers differ by magnitudes. The relation between the overall runtime (upper curves) and the time spend in the first iteration (lower curves) depends on the sample size.

of sample sizes, except for CPLEX and MOSEK that changed places for $n \geq 5000$. The CPLEX performance curve shows several non-linearities, it even has a local minimum at $n = 5000$, i.e. it takes less time to solve a problem with 5000 data points than a problem with 2000. This might be due to an automatic switch between different optimisation procedures or presolving routines that depend on the sample size.

Outer Loop Iterations The support feature machine requires to solve multiple linear programs until convergence. In the very first iteration, all columns of the data matrix $\boldsymbol{X}$ serve as input to the solver. As the number of outer loop iterations increases, more and more entries of the scaling vector $\boldsymbol{z}$ become zero and the corresponding columns of the data matrix have no influence, and, therefore, they can be discarded in a preprocessing step. Figure 4.9 shows the relation between the overall runtime (upper curves) and the runtime spent in the very first iteration (lower curves). In small sample size scenarios, this relation is large but becomes smaller as the sample size increases. For n=50000, MOSEK and CPLEX spend more than 90% of the overall runtime in the first iteration.

Dimensionality For any fixed number of data points, the runtime increases linearly to quadratically with the number of input dimensions (see Figure 4.10). Again, CPLEX and MOSEK showed the best performances, but neither was best for all data configurations. So, the choice which one to use highly depends on the actual configuration. Consistently, MATLAB's linprog and GLPK were in third and last place, respectively.

Linear Program Formulation The support feature machine may be translated into a linear program in at least two different ways (see Chapter 3.5). The number of non-zero elements of the inequality constraint matrix may largely differ depending on the sample size and the number of dimensions. Indeed, runtime measurements showed a significant difference between both formulations, however, the tendencies largely differed between solvers (see Figure 4.11). In general, it is assumed that linear programming solvers can exploit sparsity very well, and thus, sparse formulations should be favoured. As sparsity increases with the sample size, we expect the sparse version to be faster if the sample size is large enough. This is the case for three out of four solvers — GLPK, MOSEK and CPLEX — the MATLAB version does not show such a tendency. However, in real-world applications the focus is on high-dimensional small sample size scenarios, where the behaviour is even more diverse (see Figure 4.11). Here, MOSEK is the least affected by the formulation (e.g. 14 s vs. 17 s for $d = 20000$). In contrast, CPLEX takes 20 times longer on the sparse formulation than on the original one (3 s vs. 60 s).

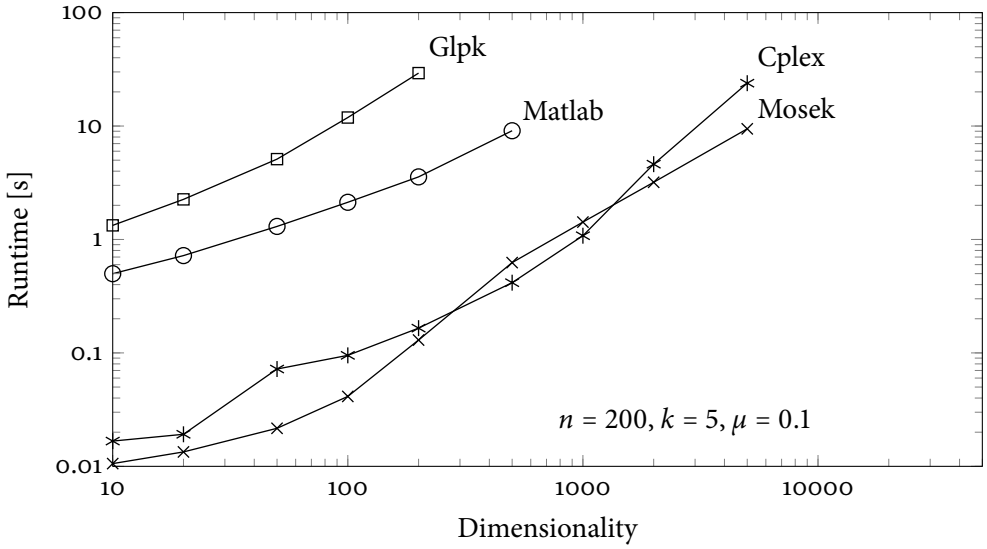


Figure 4.10: Average runtime of linear programming solvers depending on the dimensionality. The behaviour is more diverse as in the sample size dependent case (see Figure 4.9). On average, the runtimes increase approximately linearly (average slope: 0.73 (MATLAB), 1.04 (GLPK), 1.09 (MOSEK), 1.14 (CPLEX)) but may become quadratic for large dimensionalities (e.g. the slope in the last displayed interval for CPLEX is 1.79)

Intrinsic Dimensionality and Class Distance Finally, the intrinsic dimensionality k and the class distance μ both only slightly affect the runtime. The runtime is almost independent of the intrinsic dimensionality (see Figure 4.12, left) — only for an extremely small number of truly relevant features the runtime slightly decreased. Increasing the class distance parameter μ slightly reduced the runtime (see Figure 4.12, right), however, not for all solvers. All solvers seem to be unable to exploit a higher level of separability to improve the convergence speed.

Although the data was generated in a simple and canonical way, the choice of the appropriate problem formulation and the best suited optimiser is non-trivial. It mainly depends significantly on sample size and dimensionality — both are a priori known. The intrinsic dimensionality and the class distance have only a minor influence on the runtime, and they are not known in advance. Thus, the choice of the optimiser and the linear program formulation cannot be based on any of both. Figure 4.13 shows which pairs of optimiser and problem formulation perform best for a specific dataset configuration. Obviously, CPLEX performs best for large scale problems, while MOSEK seems to be better suited in low-scale problems.

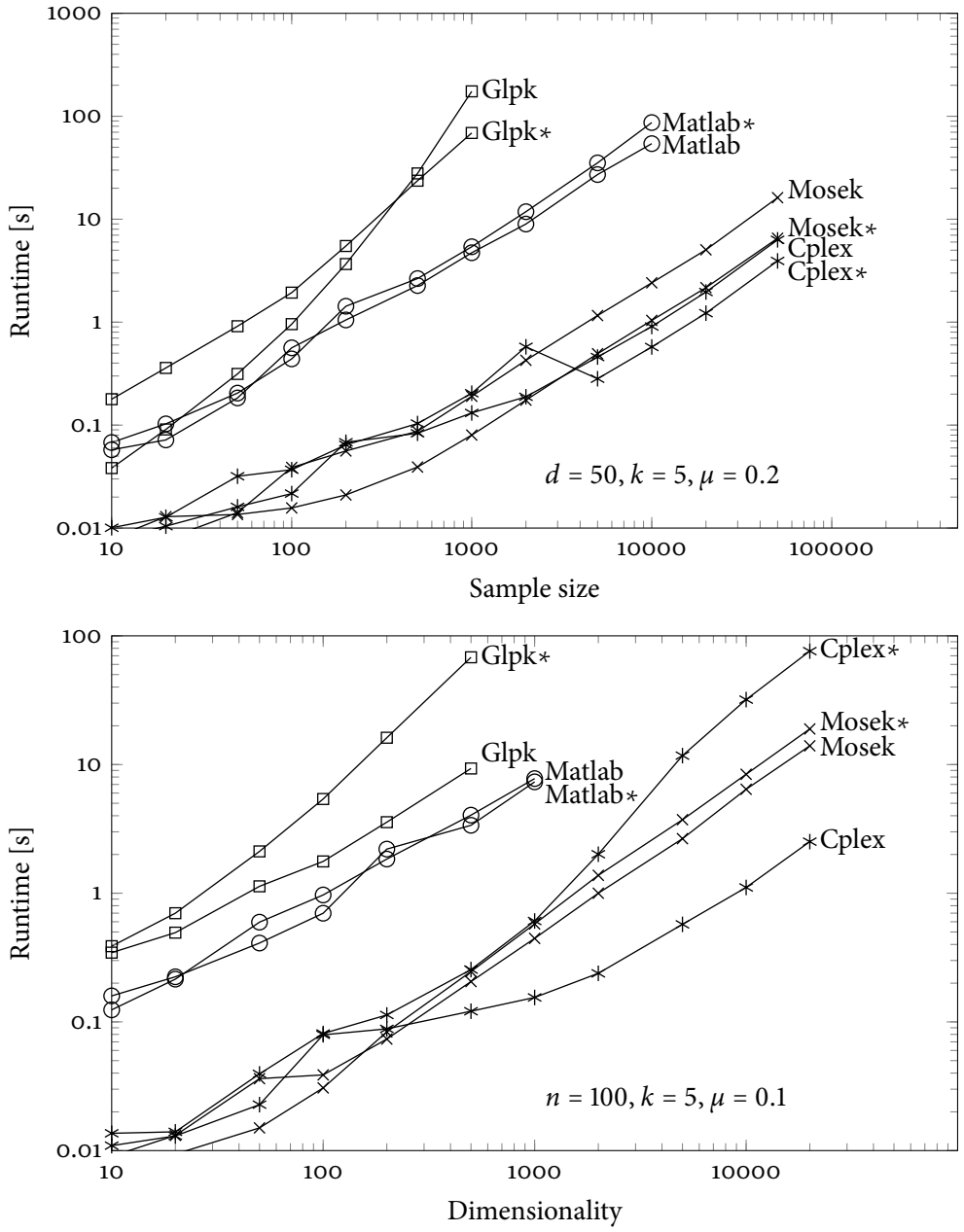


Figure 4.11: Mean runtime of linear programming solvers depending on the formulation of the linear program. In general, the second formulation — indicated by an asterisk — is sparser, i.e. the inequality matrix has less non-zero elements.

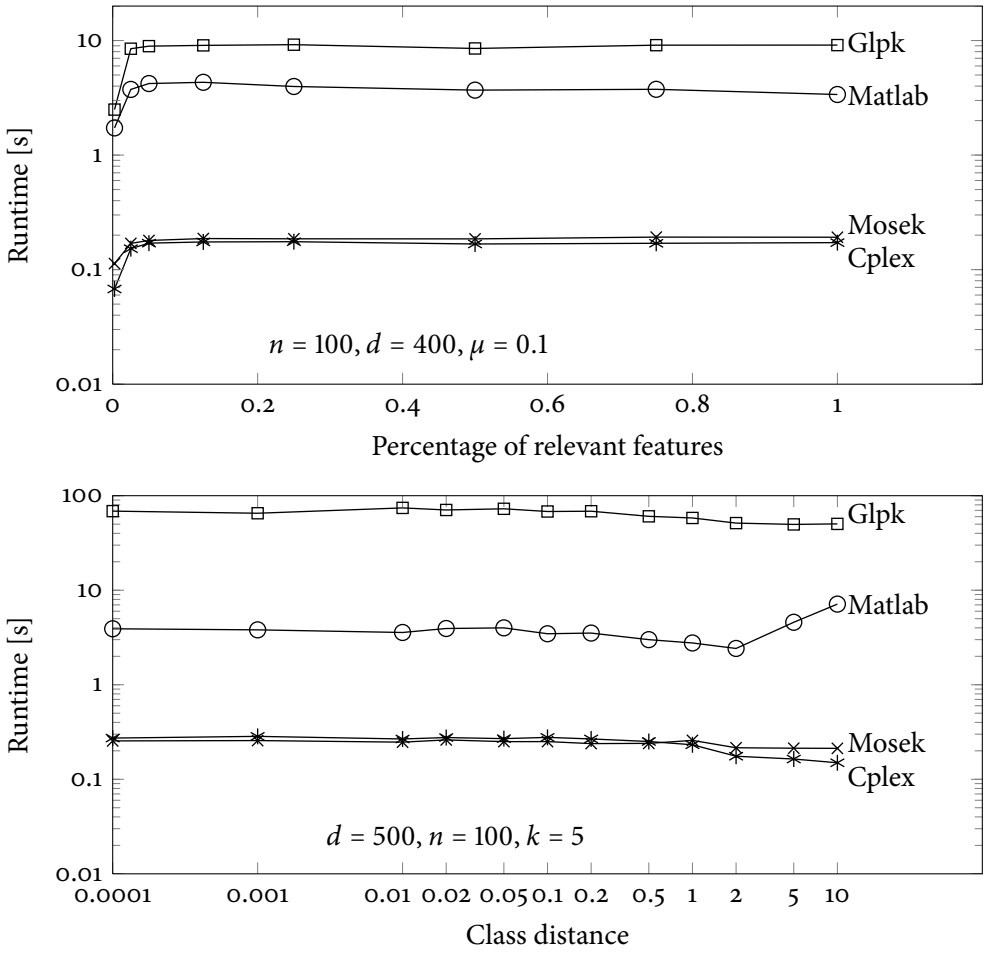


Figure 4.12: Average runtime of linear programming solvers depending on the intrinsic dimension and the distance of both classes.

Extension to Softness We conclude the runtime considerations with a comparison of hard and soft SFM. In the soft case, an individual slack variable is introduced for each data point (see Chapter 3.2): We expect the additional computational load to mainly depend on the sample size. For any fixed dimensionality the runtime difference between both approaches should increase as the sample size increases. In contrast, for any fixed sample size, the offset between both approaches should remain constant. We restricted our runtime measurements to the linear programming solver that performed best on large-size problems, which was CPLEX.

For a fixed dimensionality (see Figure 4.14, top, $d = 50$), the soft approach was always slower independent of the linear program formulation, and the runtime difference between both

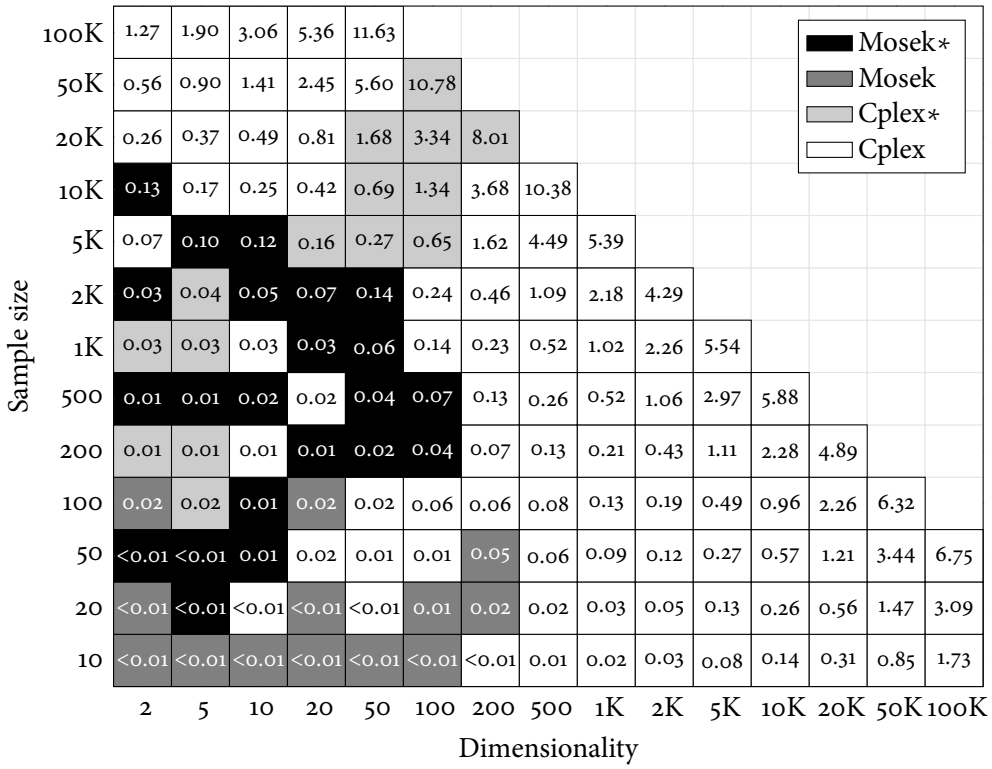


Figure 4.13: Optimal choice of optimiser and linear program formulation. For each combination of sample size and dimensionality the best optimiser configuration is shown together with the average runtime in seconds (10 runs, all iterations until convergence). Scenarios were limited to those with $n \cdot d \leq 5000000$.

approaches increased with the sample size. For $n = 10000$, a soft SFM takes almost 100 times as long as a hard-margin SFM. In contrast, for a fixed sample size (see Figure 4.14, bottom, $n = 200$) the runtimes converge such that for $d = 10000$ no difference is measurable. Again, the runtime curves are neither smooth nor strictly increasing due to solver inherent heuristics. Finally, the non-sparse formulation in connection with CPLEX is the optimal choice for high-dimensional problems, while MOSEK is generally better suited for large-sample size problems (see Figure 4.15).

The runtime measurements have shown that none of the evaluated solvers is the fastest on every dataset. The choice of the linear program formulation is crucial, and again, there is no universally optimal choice. The obtained runtime estimates were used throughout the following experiments on real-world data to always choose the best suited solver and formulation.

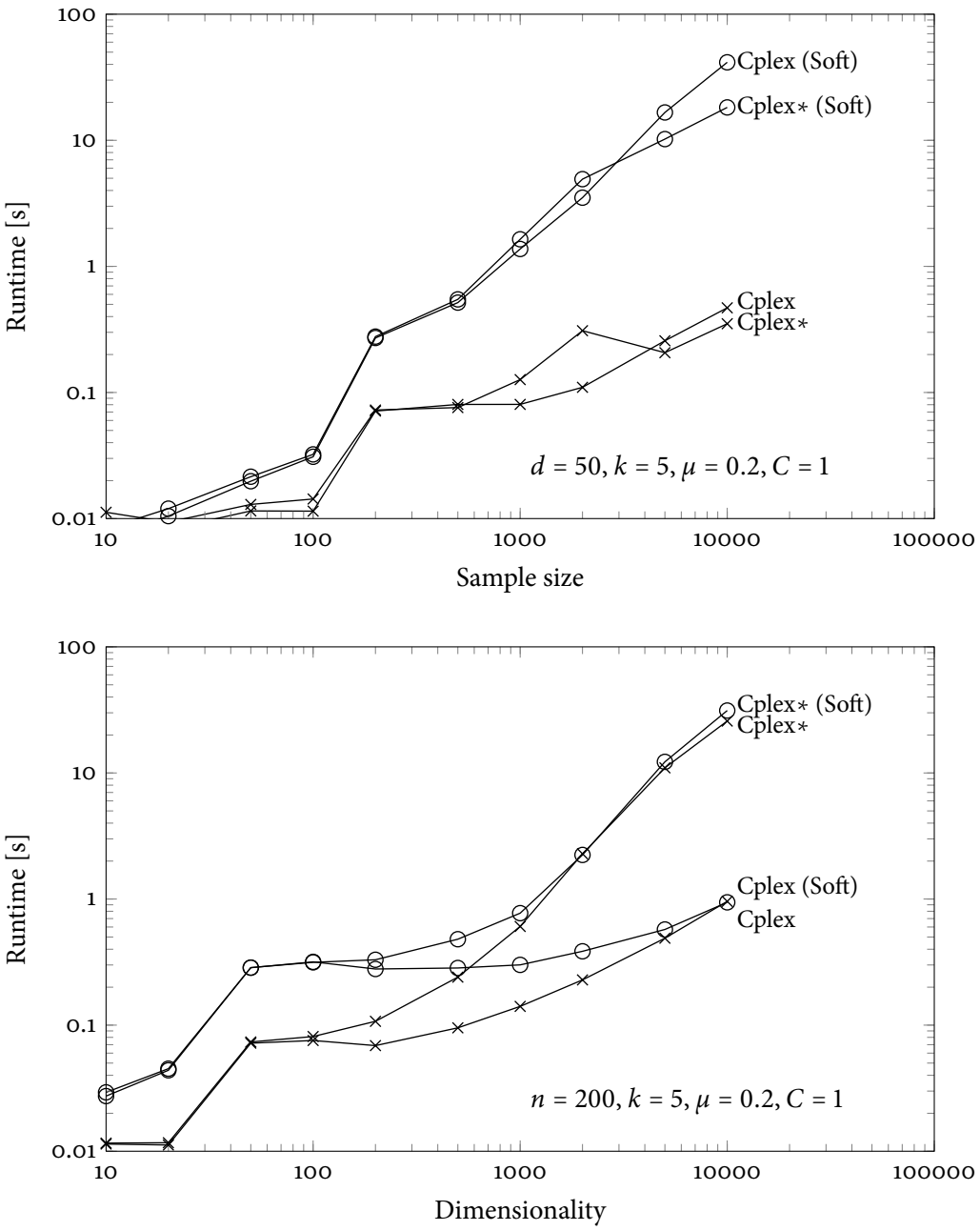


Figure 4.14: Overhead of the soft extension. For fixed dimensionality (top) the overhead increases with the sample size. In contrast, for fixed sample size (bottom) the overhead converges to zero with increasing dimensionality.

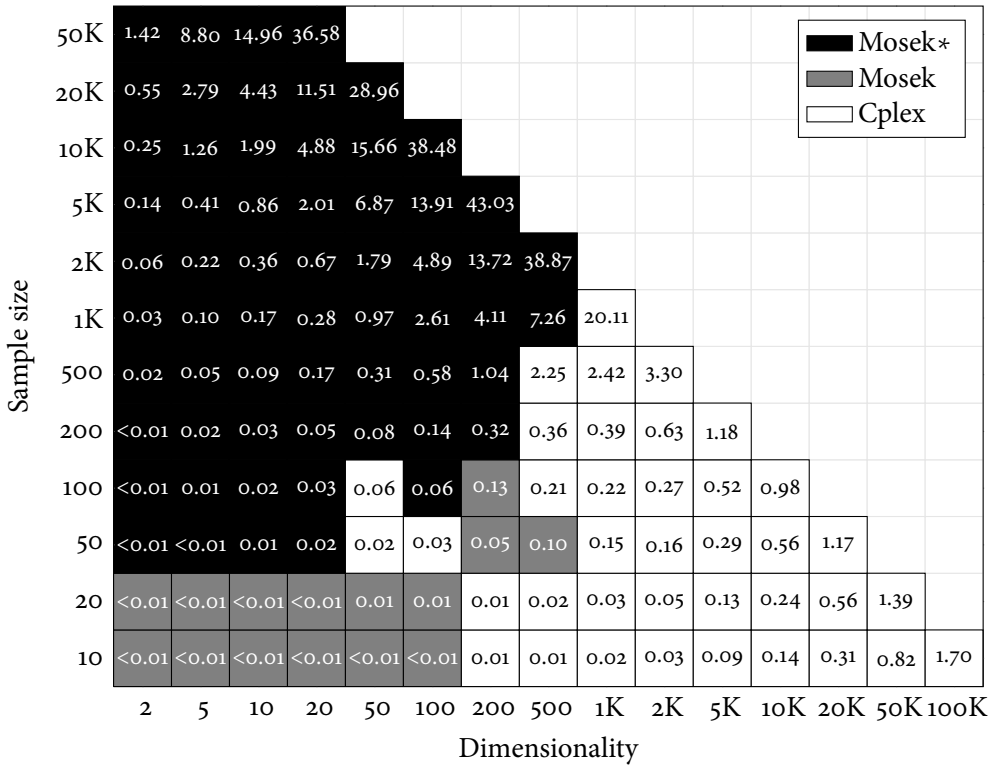


Figure 4.15: Optimal choice of optimiser and linear program formulation for the soft margin approach. For each combination of sample size and dimensionality the best optimiser configuration is shown together with the average runtime in seconds (10 runs). Scenarios were limited to those with $n \cdot d \leq 1000000$.

4.4 Evaluation on Microarray Data

Analysis of microarray data is probably one of the first biomedical applications in which machine learning methods have been widely used. In one of the first works, two types of cancer (acute myeloid leukaemia (AML) and acute lymphoblastic leukaemia (ALL)) were discriminated based on gene expression data derived from DNA microarrays using a correlation-based method [Golub et al., 1999]. As this dataset is publicly available (`www.broadinstitute.org/cancer/pub/all_aml/`), it has been used as a benchmark for numerous classification and feature selection methods [Mukherjee et al., 1998, Slonim et al., 2000, Furey et al., 2000, Guyon et al., 2002, Chapelle et al., 2002]. The dataset consists of two classes with a total of 38 training samples (27 vs. 11) and 34 test samples (20 vs. 14) with 7129 features. In the following experiments, AML data is labelled with +1 and ALL data is labelled with −1.

Correlation Analysis In a first step, we aim to reproduce the results in [Golub et al., 1999] using their method. The obtained heat map (see Figure 4.16) shows the 25 features with the largest positive and the 25 features with the largest negative correlation coefficients according to Golub's score (see Chapter 2.6.3). Our results slightly differ from those originally obtained in [Golub et al., 1999] which has several reasons. First, the publicly available dataset contains 7129 features while Golub et al. originally used no more than 6817. Second, the preprocessing and normalisation steps remain unclear — the authors claim to have normalised the log expression levels to zero mean and unit variance for each gene. However, the available data contained negative expression levels, so taking the logarithm is impossible. Our reproductions came closest to the originally published by normalising the expression levels to zero mean and unit variance (without any logarithm). The correlation coefficients are biased towards positive values, i.e. the absolute values of the largest positive correlation coefficients are larger than those of the largest negative correlation coefficients on average.

In the following, we compare the classification performance of SVM and SFM on this dataset. Further, we evaluate the capability of the SFM — with its extensions to soft separability and multiple repetitions — to identify putatively relevant features as defined by Golub's correlation-based method.

Performance of the Support Vector Machine Several methods proposed for the analysis of the *leukemia* dataset were based on the support vector machine. With a standard linear hard-margin SVM, trained on the 38 training samples, we obtained an error rate of 8.8% on the 34 test samples, i.e. three samples were misclassified. This is within the range of all previously published results (see Figure 4.21). The area under the curve (AUC) was exactly one, indicating that the orientation of the separating hyperplane obtained by the SVM was optimal and that the prediction error could further be reduced by adapting the bias. Introducing softness did not further reduce the test error. A soft-margin SVM with $C = 1$ produced a classification error of 23.5% (8/34) with an AUC of 0.99. This effect of softness is likely due to the fact that the dataset is unbalanced and introducing softness pushes the decision border towards the smaller class.

Feature Identification and Classification Performance of the Support Feature Machine The *leukemia* dataset shows strong correlations between many of the relevant features, i.e. a high degree of redundancy, and even single features are useful for prediction to some extent. Thus, we expected the data to be separable within multiple low-dimensional subspaces. We show this for the first 10 repetitions of a hard RSFM, where the number of obtained features ranges between one and four (see Figure 4.17, (a)). As discussed in Chapter 3.2, if the SFM indeed finds the optimal zero-norm minimising weight vector in each repetition, then the number of

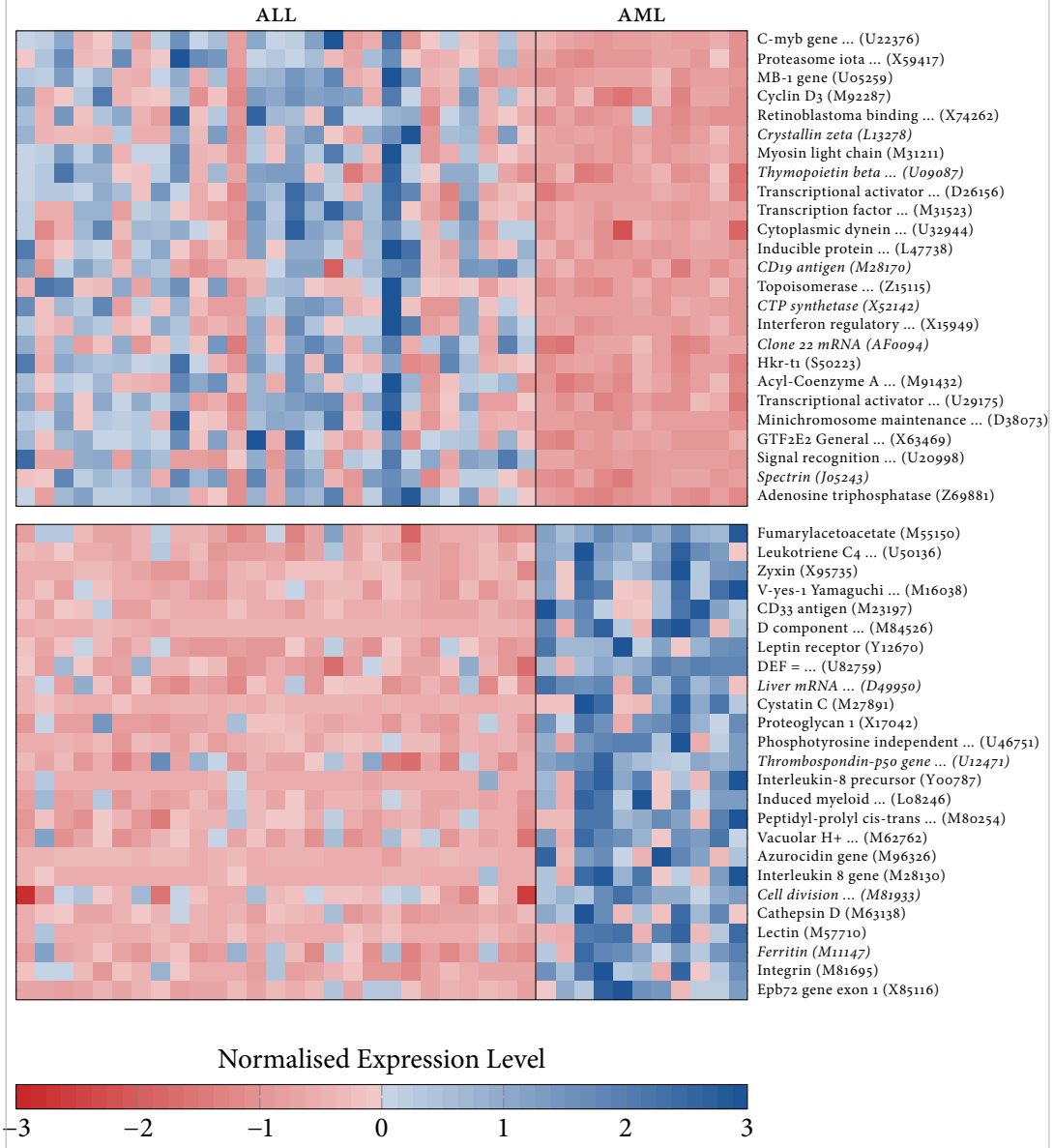


Figure 4.16: Heat map obtained according to Golub's method. Each column represents a data point; each row represents a feature. Shown are the 25 genes with largest and lowest correlation coefficient. Truncations of gene names are denoted by dots (…). Genes are typeset in *italics* if they were not present in the original feature list [Golub et al., 1999]. The deviations are due to normalisation issues and an increased number of genes with respect to the original dataset.

(a) hard SFM

Repetition	1	2	3	4	5	6	7	8	9	10
#Features	3	1	3	4	2	3	2	3	3	3
Test error (hard SFM)	0.26	0.09	0.21	0.09	0.15	0.32	0.24	0.41	0.18	0.09

(b) soft SFM

Repetition	1	2	3	4	5	6	7	8	9	10
#Features	1	1	2	1	1	1	2	3	1	2
Training error (soft SFM)	0.08	0.00	0.08	0.08	0.08	0.08	0.03	0.05	0.05	0.05
Test error (soft SFM)	0.24	0.09	0.21	0.35	0.18	0.29	0.24	0.12	0.06	0.15

Figure 4.17: Performance of the hard and soft SFM for 10 repetitions on the *leukemia* dataset. Within each repetition, the hard SFM identified one to four features that suffice to linearly separate the training data. Softness reduced the number of obtained features and led to one to three misclassified samples in the training run.

obtained features should be a monotonously increasing function of repetition. However, for reasons discussed in Section 3.2 this might in practise not always be the case. For the *leukemia* dataset, we observed no significant increase of the number of obtained features within the first 10 repetitions (see Figure 4.17, (a)). The corresponding prediction error varied significantly from repetition to repetition. A very similar behaviour was observed for a soft SFM ($C^+ = 1$, C^- chosen according to the class ratio, i.e. $C^- = C^+ \frac{n^+}{n^-}$, see Figure 4.17, (b)).

To assess whether combining several feature subsets would increase classification performance, we reordered the feature subsets returned by the RSFM according to their size, starting with the smallest feature subset, and trained an SVM on the accumulated feature sets. If the SFM indeed identifies multiple informative feature subsets, then the prediction performance should increase if several feature subsets are cumulated. This was the case for both the hard and the soft RSFM (see Figure 4.18). Moreover, overall prediction performance was better when a soft RSFM was used to identify relevant features than when a hard-margin RSFM was used — a behaviour that is expected if the data is not separable in the intrinsic feature space, i.e. the classes overlap.

To assess whether the RSFM indeed identified the putatively most relevant features within the first 10 repetitions, we compared the obtained features to those identified by Golub et al. (see Figures 4.19 and 4.20). We observed a large overlap between the obtained feature sets. For the hard SFM, 17 out of 27 features were also present in GOLUB's 50 feature list. For the soft SFM, all 15 features were present in GOLUB's 50 feature list.

(a) hard SFM

Repetition	2	5	7	1	3	6	8	9	10	4
#Features	1	2	2	3	3	3	3	3	3	4
#Cumulated features	1	3	5	8	11	14	17	20	23	27
Test error (hard SFM)	0.09	0.15	0.24	0.26	0.21	0.32	0.41	0.18	0.09	0.09
Training error (soft SVM)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Test error (soft SVM)	0.09	0.12	0.06	0.09	0.12	0.18	0.18	0.18	0.18	0.09
AUC (soft SVM)	0.97	0.97	0.99	0.99	0.99	0.99	0.98	0.99	0.98	0.99

(b) soft SFM

Repetition	1	2	4	5	6	9	3	7	10	8
#Features	1	1	1	1	1	1	2	2	2	3
#Cumulated features	1	2	3	4	5	6	8	10	12	15
Training error (soft SFM)	0.08	0.00	0.08	0.08	0.08	0.05	0.08	0.03	0.05	0.05
Test error (soft SFM)	0.24	0.09	0.35	0.18	0.29	0.06	0.21	0.24	0.15	0.12
Training error (soft SVM)	0.08	0.03	0.03	0.03	0.03	0.03	0.03	0.00	0.03	0.00
Test error (soft SVM)	0.21	0.15	0.12	0.12	0.06	0.06	0.06	0.06	0.06	0.09
AUC (soft SVM)	0.76	0.94	0.94	0.95	0.97	0.97	0.99	0.97	0.99	0.98

Figure 4.18: Classification performance of an SVM trained on accumulated feature subsets obtained with a hard (a) and a soft SFM (b).

Finally, low-dimensional feature subsets might be found due to incidental separability (see Chapter 2.5.4). However, this is not very likely. Assume a random normally distributed dataset of the same size as the *leukemia* dataset ($D = 7129$, $n = 38$). Then, $P_{2,7129,38}$ (the probability of the data being separable in just 2 dimensions) is upper bounded by 0.007, so 2-dimensional linearly separable subspaces will almost never occur by random. Further, the intrinsic dimension is probably not 7129 but much lower due to the strong correlation among many features. This, in connection with the fact that the leukemia dataset is definitely not random, provides some intuition that also subspaces of size 3 and 4 are very unlikely to occur by random. However, an accurate probability for incidental separability of the *leukemia* dataset cannot be derived.

In sum, for the *leukemia* dataset a repetitive support feature machine seems to effectively identify the most relevant genes that allow to separate the two classes. Accumulating multiple feature subsets as identified by the soft RSFM provide a smoothly increasing prediction accuracy as estimated by the SVM. However, on average both approaches — hard and soft — do not differ significantly.

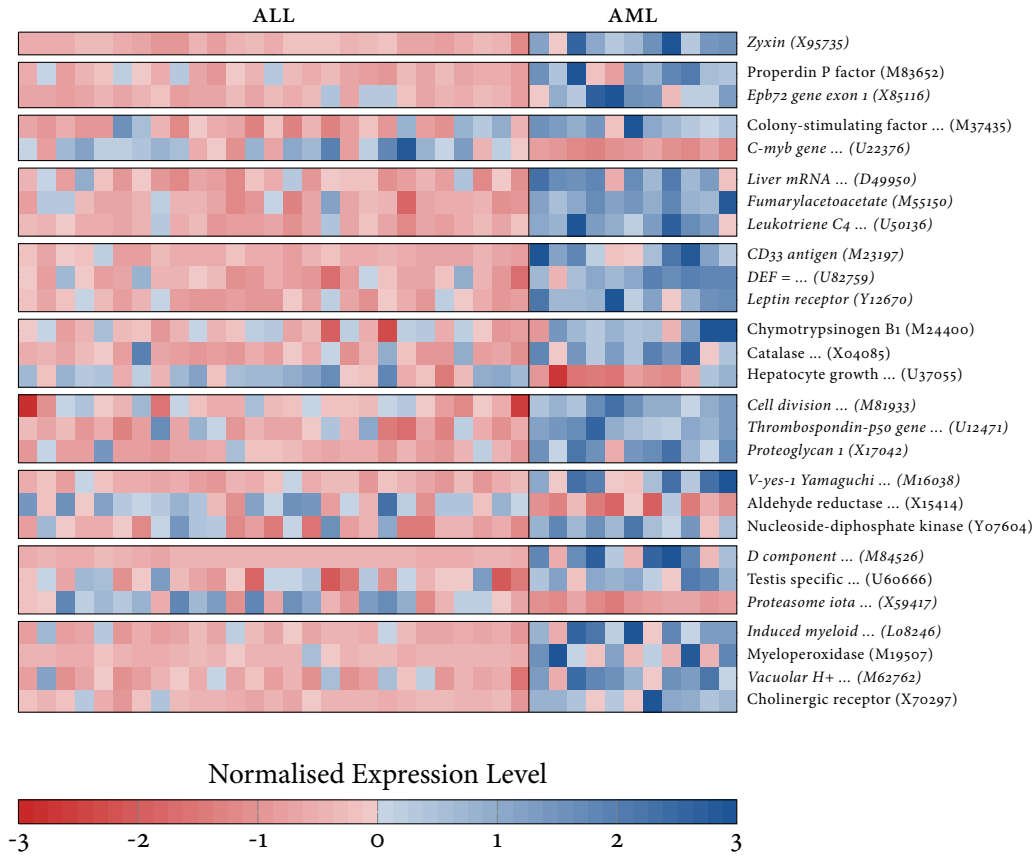


Figure 4.19: Heat maps obtained by repetitively applying the hard SFM to the *leukemia* dataset. Each column represents a data point; each row represents a feature. Each group of rows represents a feature set identified by one SFM repetition. The feature sets are sorted in ascending order according to their size. Overall, 27 genes were identified in the first 10 repetitions (see Figure 4.18). Genes are typeset in *italics* if they are listed within the top 50 features (of 7129) according to GOLUB's score. It can be seen that there is a significant overlap between both feature sets (17 out of 27 features were also in GOLUB's top 50 features list).

4.5 Conclusions

High-dimensional small sample size data has numerous unintuitive properties — distances concentrate, hubs emerge, random data points have rather deterministic than random behaviour. All these theoretical issues have practical impact, e.g. in leave-one-out cross-validation for support vector machines. The outcome of such a validation scheme is dramatically biased. In

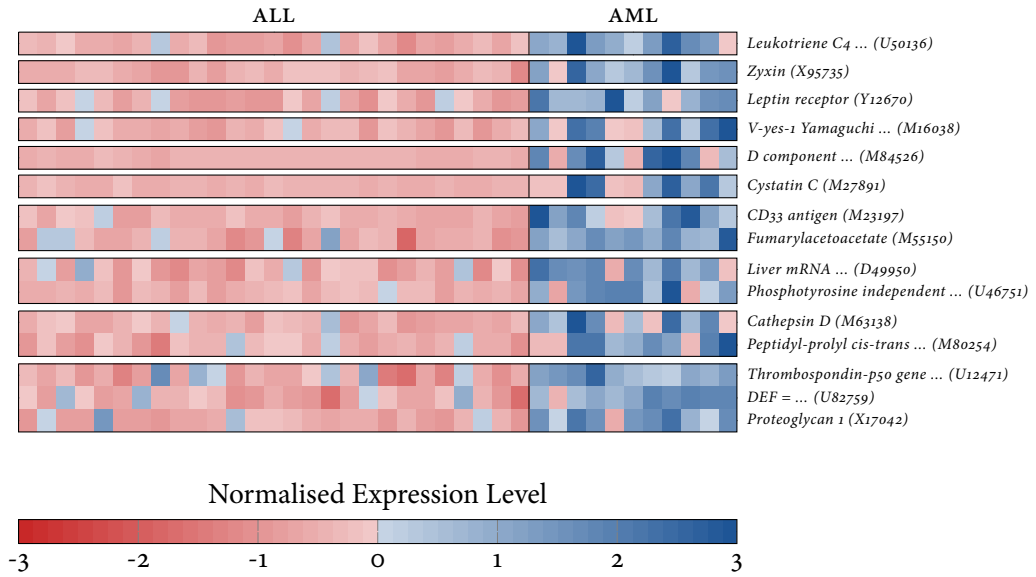


Figure 4.20: Heat maps obtained by repetitively applying the soft SFM to the *leukemia* dataset. Overall, 15 genes were identified in the first 10 repetitions (see Figure 4.18, (b)) all of which were also listed within the top 50 features according to the GOLUB's score.

Chapter 2.5.5, we proved that the error rate converges to 1 as the dimensionality goes to infinity. In this chapter, we derived characteristics to decide whether a finite random dataset behaves as if it was infinite dimensional. Similar characteristics were obtained for real two-class scenarios. Here, the error rate becomes unstable, i.e. increasing the sample size by a single pattern may cause the error rate to converge to 0 instead of 1. For soft-margin SVMs this behaviour is further amplified. With increasing softness — i.e. for small values of C — the data behaves infinite dimensional even for lower dimensionality as in the hard margin case. These experiments again motivate the necessity to limit or reduce the dimensionality of any given dataset wherever possible.

Experiments on artificial datasets show that the support feature machine very effectively discards irrelevant features and converges to the true set of features as the number of data points is increased. In comparison to the closely related SVM-based feature selection method by WESTON et al., it almost always obtains a smaller set of features which are more likely relevant ones and it provides a proper solution already in the very first iteration of the linear programming based algorithm. Additionally, it scales well — i.e. the performance degrades slowly — even if the dimensionality is increased exponentially.

Resources	Methods	Features	Accuracy
[Golub et al., 1999] and [Slonim et al., 2000]	Golub's score	50	94–100%
[Mukherjee et al., 1998]	Golub's score and linear svms	7129	97%
		999	100%
		99	100%
		49	94%
[Furey et al., 2000]	Golub's score, svm, dot-product kernel with diagonal factor	25, 250, 500, 1000	88–94%
	modified perceptron	7129	90%
[Guyon et al., 2002]	svm, recursive feature elimination	8, 16	100%
[Zhu et al., 2004]	1-norm svm	17	94%
This work	Linear hard-margin svm	7129	91%
	Repetitive hard sfm	5	94%
	Repetitive soft sfm	5	94%

Figure 4.21: The results of previous work on the *leukemia* dataset and the results we obtained with the sfm. All methods achieve accuracies between 88% and 100%, however, the number of included or obtained features differs significantly.

In any practical implementation of an sfm the choice of the linear programming solver is crucial. We evaluated the runtime performance of the sfm based on four different solvers and found the commercial toolboxes mosek and cplex to outperform matlab and glpk. The runtime increases linearly both with the number of features and the number of samples. However, for a particular scenario the runtimes of the alternative linear program formulations — standard or sparse — may differ by orders of magnitude. For the basic sfm we observe cplex to be better suited for large-scale problems (large dimensionality or large sample size) while mosek is better suited for small-scale problems. For the soft sfm, we found cplex to be better suited for high-dimensional problems, while mosek is better suited for large sample-size problems. For both sfm variants, we provided look-up tables to choose the best suited optimiser and linear program formulation.

Finally, we evaluated the sfm on a real-world dataset. The *leukemia* microarray dataset is a well-known example of high-dimensional small sample size data and has been used by many authors for benchmarking machine learning methods. Here, we found the repetitive

SFM — both hard and soft — to effectively filter out relevant features. The test error was in the same range as other state-of-the-art methods, while the number of obtained features was significantly lower. Five features were sufficient to achieve a prediction accuracy of 94%. Besides, we empirically verified that a soft SFM trades off a hard SFM and correlation based feature selection — a repetitive soft SFM successively selects feature sets with a large GOLUB's score.

They buy the place up with gold and still they have all the gold. Swindle in it somewhere. Piled up in cities, worn away age after age. Pyramids *in sand. Built on bread and onions. Slaves Chinese wall. Babylon. Big stones left. Round towers. Rest rubble, sprawling suburbs, jerrybuilt. Kerwan's mushroom houses built of breeze. Shelter, for the night.*
No-one is anything.

«Ulysses», James Joyce

5 Image Processing Excursus: The Gaussian Pyramid for Illumination Correction

The previous chapters were related to machine learning and feature selection with the aim to define a method for finding the minimum number of features that are necessary to separate two classes. However, prior to any machine learning method, it is often necessary to apply further preprocessing techniques to remove data acquisition artefacts and to improve the signal-to-noise ratio. In this chapter, we introduce such a preprocessing technique specifically for image data.

Digital images in many ways suffer from deficiencies of the hardware that was used for capturing the image, including sensor, lenses, and illumination. Even with a perfect sensor, a lens without aberrations and a perfectly homogeneous illumination, the image may still show an intensity falloff towards the corners of the image due to natural vignetting. In general, the illumination inhomogeneity is much more complex [Aggarwal et al., 2001b] and cannot be described with a simple model.

Illumination correction methods have been addressed by many authors, often with respect to a specific illumination artefact. Vignetting correction methods have been proposed e.g. in [Zheng et al., 2009] and [Kim and Pollefeys, 2008]. In magnetic resonance imaging the correction components may be modelled as combinations of smoothly varying basis functions [Likar et al., 2001]. In face recognition, methods for illumination compensation are used as a preprocessing step to find an invariant face representation [Adini et al., 1997]. An overview of stitching and blending methods can be found in [Levin et al., 2004].

We propose a novel illumination correction method based on low-pass filtering using Gaussian pyramids extended by an appropriate image extrapolation to avoid boundary artefacts.

This extrapolation step is essentially different to all commonly used techniques. Originally, it was designed to reduce visible image transitions when stitching images to form larger mosaics [Klement et al., 2011]. Such methods are of great importance in virtual material design and rapid prototyping, where realistically looking scenes are rendered in software to avoid costly physical models. However, we found that our method — especially the way in which the boundary of an image is handled in the filtering step — is also suited for downsampling 3-dimensional fMRI data. This will be further analysed in Chapter 6.4.4.

This chapter is organised as follows. First, we introduce a Gaussian pyramid-based framework to remove illumination gradients from intensity images. Second, we apply the framework to artificial and real-world images to show how the proper choice of the boundary extension relates to intensity artefacts.

5.1 Illumination Correction Framework

For more than two decades, pyramid methods have been used for image processing tasks such as image enhancement, compression, interpolation and extrapolation of missing image data and numerous others [Ogden et al., 1985]. Given a gray-valued input image with intensity values $G_0(x, y)$ at discrete locations $x \in [0, m-1]$ and $y \in [0, n-1]$ the Gaussian pyramid is an efficient data structure for spectral decomposition: For each level i the image $G_i(x, y)$ is low-pass filtered and downsampled by a factor of two to produce the image $G_{i+1}(x, y)$. This is commonly referred to as the *reduce* operation, i.e.

$$G_{i+1} = \text{reduce}(G_i) = (\downarrow 2)(h * G_i) \quad \text{with} \quad (\downarrow 2) f(x, y) = f(2x, 2y)$$

where the filter operator $*$ is defined as

$$(f * g)(x, y) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} g(i, j) f(x - i, y - j)\,. \tag{5.1}$$

The *reduce* operation is repeated up to a certain level l. Then, the *expand* operation is applied successively to get an image with the same size as G_0, but containing only the low frequency components of G_l:

$$G'_{i-1} = \text{expand}(\text{G}_\text{i}) = 4\, h * ((\uparrow 2) G_i)\,.$$

The weighting function h — the *generating kernel* — is commonly chosen to be a 5-by-5 binomial filter to approximate the Gaussian. This method for computing the low frequency components of the input image is more efficient than direct convolution with a large filter but also more efficient than using standard *fast Fourier transform* (fft) [Adelson et al., 1984]. In practise, as

images are of finite size the filtering operation (5.1) would be undefined for boundary pixels such that the image needs to be extended by two pixels — either virtually or explicitly — in each direction when using 5-by-5 filters. Commonly, the pixels outside the image are set to a constant value or to the value of the nearest boundary pixel (*replicate boundary*) or are computed by assuming a periodic image (*circular boundary*). The following general framework allows to define arbitrary boundary conditions: Given the input image $f(x, y)$ with $x \in [0, m-1]$ and $y \in [0, n-1]$ we define the extended image $f'(x, y)$ for $x \in [-2, m+1]$ and $y \in [-2, n+1]$. The set of pixels

$$\mathcal{P}_{x,y} = \left\{ \left(p_i^x, p_i^y \right) \right\}_{i=0}^{|\mathcal{P}_{x,y}|-1}$$

contains all those pixels that influence the intensity at (x, y) in the extended area of the image. The function $g_{\mathcal{P}_{x,y}}(x, y)$ defines how to calculate the intensity at (x, y) from the intensities of the set of pixels $\mathcal{P}_{x,y}$. Thus,

$$f'(x, y) = \begin{cases} f(x, y) & \text{if } x \in [0, m-1] \text{ and } y \in [0, n-1] \\ g_{\mathcal{P}_{x,y}}(x, y) & \text{otherwise.} \end{cases}$$

For convenience, we use $g(x, y)$ instead of $g_{\mathcal{P}_{x,y}}(x, y)$. Assuming a *replicate boundary*, the intensity values of pixels outside of the image equal those of the nearest boundary pixel:

$$\mathcal{P}_{x,y}^{\text{rep}} = \left\{ \left(p_0^x, p_0^y \right) \;\middle|\; \begin{array}{l} p_0^x = \min(\max(x, 0), m-1), \\ p_0^y = \min(\max(y, 0), n-1) \end{array} \right\}.$$

Thus, $\left|\mathcal{P}_{x,y}^{\text{rep}}\right| = 1$ for all x and y, and $g^{\text{rep}}(x, y) = f(p_0^x, p_0^y)$. Analogously, a *circular boundary* condition corresponds to

$$\mathcal{P}_{x,y}^{\text{circ}} = \left\{ \left(p_0^x, p_0^y \right) \;\middle|\; \begin{array}{l} p_0^x = x \bmod m, \\ p_0^y = y \bmod n \end{array} \right\}.$$

Again, $|\mathcal{P}_{x,y}^{\text{circ}}| = 1$ for all x and y and $g^{\text{circ}}(x, y) = f(p_0^x, p_0^y)$. Both assumptions — replicate and circular — are inapproriate when modelling illumination gradients. When compensating for vignetting, the application of replicate or circular boundary assumptions overestimates the intensity gradient at the boundary. We propose to use extrapolation methods that allow to model smoothly continuing intensity gradients beyond the image boundary. With the above framework, *linear extrapolation* would be achieved by:

$$\mathcal{P}_{x,y}^{\text{lin}} = \left\{ \left(p_0^x, p_0^y \right), \left(p_1^x, p_1^y \right) \;\middle|\; \begin{array}{l} p_0^x = \min(\max(x, 0), m-1),\ p_0^y = \min(\max(y, 0), n-1) \\ p_1^x = \min(\max(x, 1), m-2),\ p_1^y = \min(\max(y, 1), n-2) \end{array} \right\}$$

and

$$g^{\text{lin}}(x,y) = f\left(p_0^x, p_0^y\right) + \left(f\left(p_0^x, p_0^y\right) - f\left(p_1^x, p_1^y\right)\right) \cdot \left\| \begin{pmatrix} x - p_0^x \\ y - p_0^y \end{pmatrix} \right\|_1 .$$

A more stable extrapolation involves least-squares regression. Assume $\mathcal{P}_{x,y} = \left\{\left(p_i^x, p_i^y\right)\right\}$ to contain a 5-by-5 block of pixels within the image bounds that is closest to the point (x, y). By

minimising $\quad \|\mathbf{A}\beta - \mathbf{b}\|_2$

with $\quad \mathbf{A} = \begin{pmatrix} p_0^x & p_0^y & (p_0^x)^2 & (p_0^y)^2 & p_0^x p_0^y & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{24}^x & p_{24}^y & (p_{24}^x)^2 & (p_{24}^y)^2 & p_{24}^x p_{24}^y & 1 \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} f(p_0^x, p_0^y) \\ \vdots \\ f(p_{24}^x, p_{24}^y) \end{pmatrix}$

we obtain the coefficients of a two-dimensional second order polynomial, modelling the intensity in the region $\mathcal{P}_{x,y}$ with minimum error. Thus, the intensity at (x, y) is approximated by

$$g(x,y) = \begin{pmatrix} x & y & x^2 & y^2 & xy & 1 \end{pmatrix} \cdot \beta \quad \text{with} \quad \beta = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}\,.$$

Alternative regression approaches may be applied. However, the complexity of a regression model, i.e. the number of parameters to be estimated, should be low, as illumination gradients are by definition smooth functions. The choice of the neighbourhood size is a trade-off between stability and runtime, but, as our experiments show, 5-by-5 blocks are sufficient to remove almost all visible boundary artefacts. The computational complexity of the regression method is no crucial factor, as extrapolation is done only for a low number of boundary pixels. So, other operations, such as filtering of the whole image, dominate the complexity of the illumination correction method. In the overall algorithm (see Figure 5.1), extrapolation of boundary pixels is used in steps (4) and (13).

In the downsampling loop (2–8), the image is first extended by two rows and columns in each direction and then the intensity values of these pixels are determined by extrapolation. After filtering, the image is cropped to the original size. In the upsampling loop (10–15), the image is first filtered and then the intensity values of the first and last two rows and columns are recalculated from the inner image by extrapolation. Finally, the low-frequency image is subtracted from the original image and the mean intensity of the input image is added to obtain the final image with the proper intensity level.

The removal of low frequent image components with the aforementioned method may alter the image histogram, especially if the input image contained saturated pixels. Thus, further postprocessing steps, such as histogram matching [Rolland et al., 2000], may be required to improve the visual impression.

Input : Intensity image $f(x, y)$, pyramid depth l
Output: Corrected image $f'(x, y)$

$G_0 \leftarrow f$;
for $i \leftarrow 0$ **to** l **do**
    Extend G_i;
    Extrapolate image G_i;
    Filter image, i.e. $G_i \leftarrow h * G_i$;
    Crop image G_i to original size;
    Downsample, i.e. $G_{i+1} \leftarrow (\downarrow 2)(G_i)$;
end
$G'_l \leftarrow G_l$;
for $i \leftarrow l$ **down to** 1 **do**
    Upsample, i.e. $G'_{i-1} \leftarrow (\uparrow 2)G'_i$;
    Filter image, i.e. $G'_{i-1} \leftarrow h * G'_{i-1}$;
    Extrapolate image G'_{i-1};
    Scale image, i.e. $G'_{i-1} \leftarrow 4 \cdot G'_{i-1}$;
end
Calculate mean intensity, i.e. $\overline{f} = \frac{\sum_x \sum_y f(x,y)}{nm}$;
$f' \leftarrow f - G'_0 + \overline{f}$;

Figure 5.1: Overall illumination correction algorithm.

With a minor change, the above framework applies to arbitrary non-rectangular images where the input image $f(x, y)$ is defined for a set of pixels $I \subseteq [0, m-1] \times [n-1]$. We define the extended image by:

$$f'(x, y) = \begin{cases} f(x, y) & \text{if } (x, y) \in I \\ g_{\mathcal{P}_{x,y}}(x, y) & \text{otherwise.} \end{cases} \tag{5.2}$$

Here, $\mathcal{P}_{x,y}$ contains those pixels with minimum Euclidean distance to the query point.

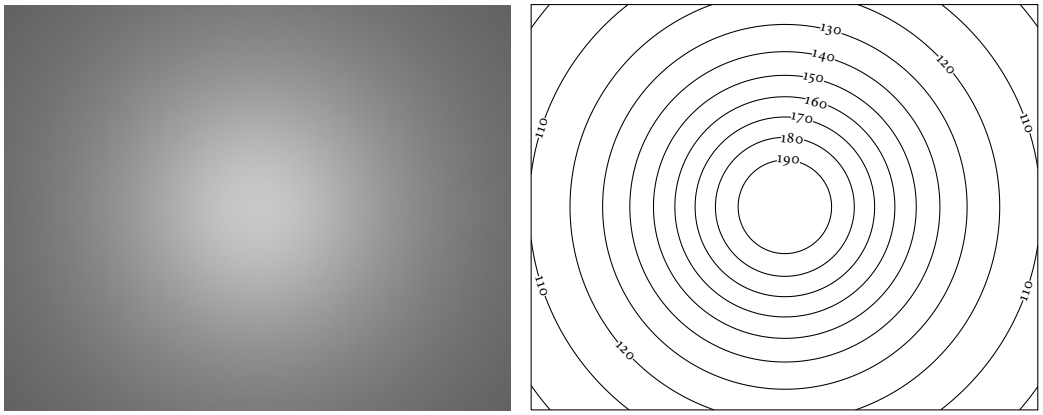
The above method can analogously be applied to 3-dimensional volumetric data — $f(x, y)$ becomes $f(x, y, z)$, the generating kernel is now a 5-by-5-by-5 binomial filter, and so on. Thus it can be used for volumetric medical imaging data such as fMRI data to remove illumination gradients or for downsampling.

5.2 Evaluation on Artificial and Real-World Data

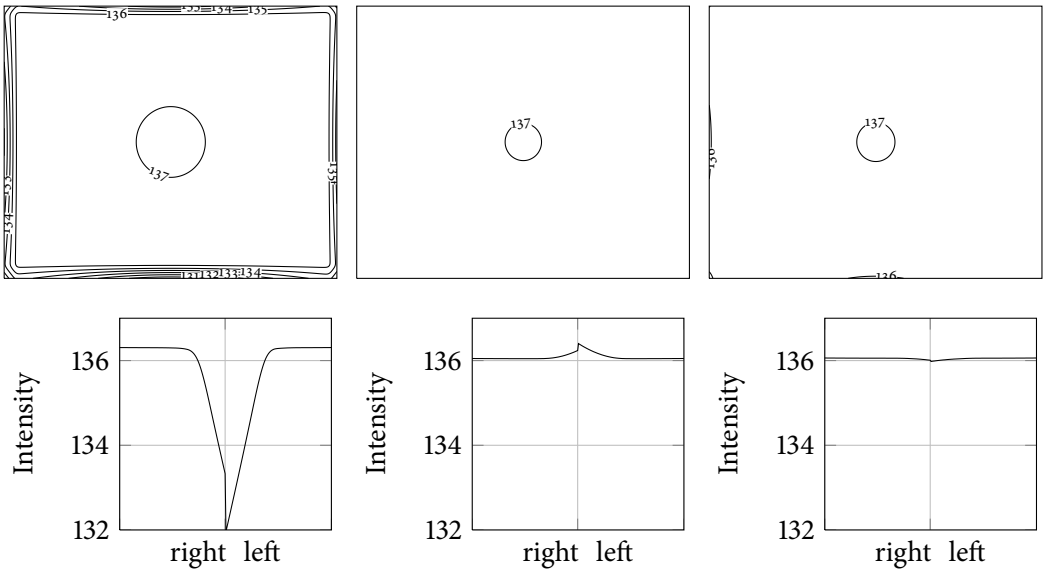
In the following experiments, we used gray-valued and RGB images with intensity values ranging from 0 to 255. All operations where done in double-precision arithmetic to avoid discretisation artefacts.

We start with an artificial image that contains no fine-grained texture but a smooth intensity gradient as in natural vignetting according to the *cosine fourth law*. Thus, the intensity I decreases with $I(\alpha) = I_0 \cdot \cos^4\alpha$ where I_0 is the maximum intensity and α the angle of the incident light. Based on a full format sensor (36 × 24 mm), a fish-eye lens with a focal length of 10 mm and disregarding any further lens aberrations we obtain the image in Figure 5.2, (a) ($I_0 = 200$, 1280 × 1024 px). Perfect illumination correction would result in a completely homogeneous image. The choice of the boundary condition does not affect the centre of the image (see Figure 5.2, (b)–(d)), which shows a slight gradient even after correction. However, the intensities at the image boundaries differ significantly. A Gaussian pyramid-based illumination correction (5 levels) with the replicate condition (see Figure 5.2, (b)) causes an intensity drop at the boundary visible as linear isophotes (contours of equal luminance). The replicate condition is outperformed by linear and polynomial extrapolation (see Figure 5.2, (c) and (d), respectively), both reduce the intensity change to less than 1 at the boundary. Thus, the correction error is below the discretisation error for 8-bit images.

The very same artefact is visible in real-world textures, as they are used in virtual material design for rapid prototyping (see Figure 5.3). The images were acquired with an industrial colour camera (Baumer TXG14c, 1392 × 1040 px) and show clearly visible intensity falloffs towards the image corners. These falloffs become particularly visible in image mosaics (see Figure 5.3, left column). We applied our illumination correction method to each colour channel individually and compared the results of the replicate boundary condition and the polynomial boundary extrapolation. Additionally, we matched the histograms of the corrected images to the ones obtained from the central quarter of the respective input images. After correction, the low-frequent intensity gradients are successfully removed. However, as in the previous experiment, the replicate boundary condition introduces edge parallel intensity falloffs (see Figure 5.3, middle column). They are almost invisible within a single image, but obvious in image mosaics by dark vertical and horizontal lines. With a polynomial boundary extrapolation no darkening is visible (see Figure 5.3, right column). Still, the obtained image mosaics are not visually pleasing. Inhomogeneities and artefacts such as repetitive texture patterns, reflections, subtle blurring towards the image corners due to lens deficiencies, and saturated pixels cause an artificial look of the stitched image mosaics. This could be improved, e.g. by rearranging texture patches from the original image. In [Efros and Freeman, 2001], each patch is chosen



(a) Input image (natural vignetting, cosine fourth law)



(b) Replicate boundary condition (c) Linear extrapolation (d) Polynomial extrapolation

Figure 5.2: Illumination correction without texture. The input image (a) visualises natural vignetting according to the *cosine fourth law*; isophotes form concentric circles. The isophotes of the corrected images (second row) show a clear intensity artefact at the image borders for the replicate condition (b). For both extrapolation methods (c,d) this is significantly reduced. The intensity across the middle row of each image (bottom row, 128 pixels from both sides) shows an intensity peak of 4.35 (a), 0.36 (c), and 0.08 (d), respectively.

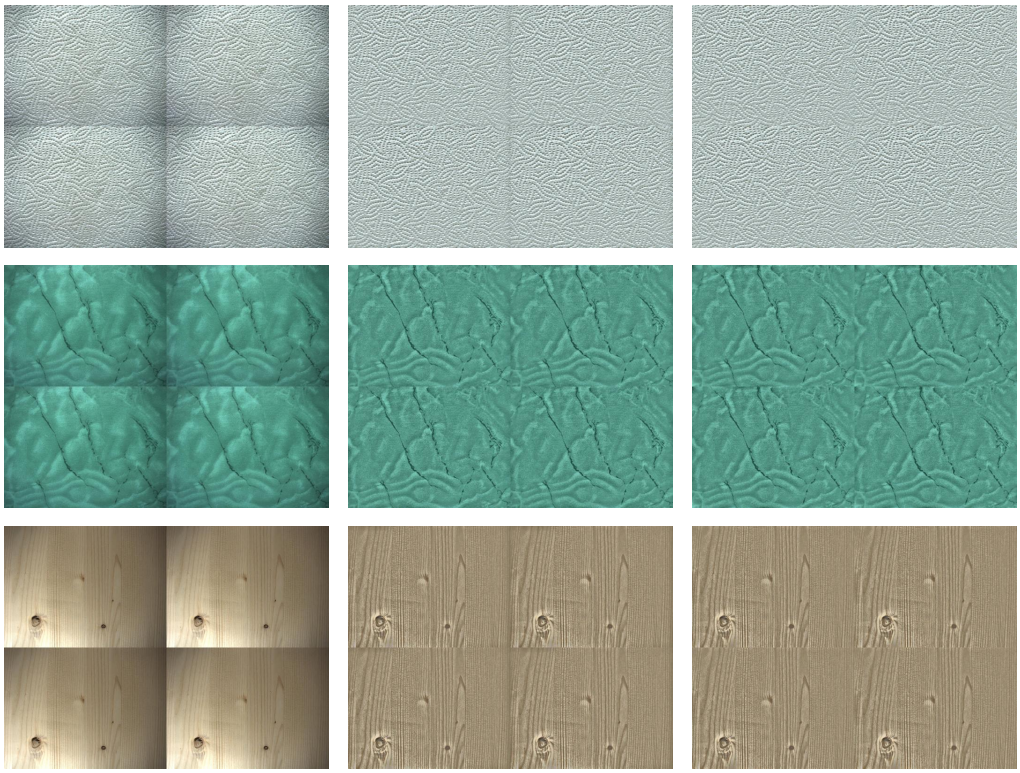


Figure 5.3: Real world colour textures. The performance of the illumination correction is shown for some real-world textured materials (from top to bottom: paper towel, green linoleum, wood; 1392 × 1040 px, 5 levels) in 2-by-2 image mosaics. Without illumination correction (left column) vignetting is most obvious. The middle column shows image mosaics after correction with a replicate boundary condition. Obviously, the correction using polynomial boundary extrapolation (right column) gives the best results.

to optimise an overlapping region with the so far synthesised texture, and, finally, the blockiness of the boundary is reduced by finding the minimum cost path within this region.

5.3 Conclusions

We introduced an illumination correction framework that minimises boundary artefacts by extrapolating the image boundary in each filtering step. The assumption of a smooth intensity gradient that continues beyond the image boundary motivates that linear and polynomial extrapolation are better suited than the common replicate or circular conditions. Our framework

allows arbitrary extrapolation functions and arbitrary shaped images. However, the complexity of the extrapolation function should be kept low as intensity gradients are by definition low frequent and slowly varying.

In practise, the standard replicate boundary condition introduces edge parallel darkening artefacts. Simple extrapolation schemes — linear or polynomial — reduced this artefact below the discretisation level for 8 bit images. In real-world textures, the method effectively removes intensity gradients, however, postprocessing methods are necessary to obtain visually pleasing large scale textures.

When he had taken aim he let fly, and his arrow pierced every one of the handle-holes of the axes from the first onwards till it had gone right through them, and into the outer courtyard.

«Odyssey», Homer
translated by Samuel Butler

6 Mindreading: Classification and Feature Selection for Brain Activity Data

Brain-computer interfaces, neuromarketing, lie detection, and mindreading based on electrophysiological or neuroimaging data have been established in cognitive neuroscience research during the last years (see [Haynes, 2011] for a recent review) but are still far from being considered for widespread application. Several studies — some based on electroencephalography (eeg) others on functional magnetic resonance imaging (fmri) or even invasive methods — have shown that brain reading, i.e. the prediction of a certain behaviour from neural activity, is feasible to some extent. The challenges range from making costly and monstrous imaging devices affordable to information theoretic considerations and data analysis techniques. Surprisingly, although the human brain is thought to process information in highly complex neural networks, multidimensional machine learning methods have only recently been introduced for analysis. In this comparatively new field of research, the sfm, being an intrinsically multidimensional method, might provide new insight in how information in general is processed in the human brain.

So far, our focus was to qualify the sfm as a method to identify the least number of features to discriminate two classes. Now, we aim to show that the sfm can reveal meaningful results from neuroimaging data. In particular, the sfm will be used to address five questions:

1. *How well may brain states be discriminated?* Given solely brain activity data of a person in a particular mental state — is it possible to tell whether the person felt joy or sadness? Traditionally, univariate statistics have been the method of choice, i.e. individual brain locations were analysed whether they are suited to decode the mental state. Only recently, multidimensional pattern recognition has become in vogue and is still at its infancy

[Haynes and Rees, 2006]. Affective states are supposed to constitute complex patterns whereas motor control tasks can already be predicted accurately on a single-trial basis with univariate methods [Dehaene et al., 1998].

2. *How many voxels are at least required to discriminate functional brain states and what is the most informative voxel set?* In neuroimaging, one is very often not only interested whether two brain states can be discriminated, but also where the most informative voxels are located. Previous studies that used multivariate decoding to compare structural or functional maps have often reduced these maps in an initial univariate step to those voxels that meet certain criteria, such as stability across trials [Mitchell et al., 2008]. However, such univariate criteria do not capture multidimensional dependencies and likely overestimate the true number of voxels required to discriminate between brain states, while voxels that carry significant information only in combination with others might be missed. In contrast, multivariate feature selection methods such as the sfm combine feature selection and classification and determine the smallest set of voxels that discriminates between two brain states.

3. *How many voxels carry relevant information and what are the least informative voxels?* The human brain is characterised by its organisation into multiple hierarchical and parallel neural networks, many of which carry similar information. Due to the high-dimensional small sample size character of neuroimaging data the exact number of voxels that carry information alone or in connection with others cannot be determined as the number of data points is usually insufficient to capture all sources of variance and to accurately describe the decision border. However, the proportion of informative voxels may be estimated with some simplifications and heuristics. This should allow to capture tendencies and to compare different types of functional and pathological brain states.

4. *How do brain states evolve over time?* Brain states are no instantaneous events but evolve over time. This behaviour should be reflected both in how well these states may be discriminated and in the number of informative voxels.

5. *What is the minimal resolution for discriminating brain states and how does the performance degrade with the resolution?* The analysis of brain activity data involves many preprocessing steps, such as registration, normalisation and smoothing. Being essential for comparing activity across participants, these steps reduce the possibility to recognise fine-grained patterns. However, prominent and global patterns should be preserved even if the data is downsampled to a lower resolution.

The following chapter is organised as follows. First, we give an overview of what constitutes

brain activity data and how it is commonly processed. We summarise previous machine learning based methods for the analysis of brain activity data and motivate why an SFM-based approach is promising. Second, we analyse whole brain activity with a mass univariate method, a very simple randomised support vector approach, and the SFM. In all analyses, we focus on cross-participant prediction. We aim to identify informative activity patterns and to predict the brain state for participants that were not included in the training data. For a simple motor task — pressing a button with either the left or right thumb — we show that an SFM-based approach identifies similar brain regions to be relevant as classical mass univariate methods do. Additionally, we show that our approach is suited to identify the number of informative voxels and the number of voxels that are non-informative. However, univariate statistics are by definition inappropriate for analysing complex multidimensional patterns as they are supposed to occur in emotional brain states. Finally, we investigate the performance of the support feature machine in discriminating emotional brain states. We not only show how well the five fundamental emotions — joy, anger, disgust, fear and sadness — may be discriminated, but also that affective information is encoded in whole brain patterns. Moreover, we show that emotions spread over time such that finally most of the brain is involved.

6.1 Data and Preprocessing

Functional magnetic resonance imaging (fMRI) measures neural activity by the blood-oxygen-level dependent (BOLD) activity. The standard fMRI preprocessing steps are mentioned here to illustrate why the analysis of brain states — especially across participants — is extremely complex. Due to anatomical differences between participants, technical restrictions of magnetic resonance imaging and information theoretic limitations, we expect brain states to be even better separable than what we inferred from the data. Small-sized activity patterns might either be invisible to fMRI scanners or they may get blurred during the preprocessing.

Common fMRI scanners acquire a 3-dimensional image in a series of 2-dimensional slices. Thus, first and last slice have a time delay of almost the repetition time TR (e.g. 3 seconds). This is accounted for by *slice acquisition time correction* — implemented by shifting and interpolating the data at consistent timestamps. Next, *motion correction* realigns the whole brain volume to a common reference by rigid body transformations. *Spatial normalisation* maps the brain volume to a standard brain and *spatial filtering* improves the signal to noise ratio but may also blur fine grained activation regions. *Grand mean scaling* refers to a normalisation method that divides each voxel value by the mean of all values within the same scanning session. The obtained sequence of brain activity volumes may be further condensed as *beta images* by estimating the activity change across a whole trial. The volumetric data is linearised to fit into standard data

analysis and machine learning frameworks. For visualisation, the activity data is mapped onto a standardised brain anatomy to allow for spatial interpretation of the data. The outcome may then be visualised as brain slices, 2-dimensional projections or directly as a rotatable object in an interactive framework. Currently, the standard brain template is the ICBM152, being the average of 152 individual MRI scans.

Image preprocessing and BOLD activity estimation was conducted with SPM5 — a free software package for the analysis of brain imaging data sequences (Wellcome Department of Imaging Neuroscience, London, UK). Brain activities were visualised based on the BrainNet Viewer (National Key Laboratory of Cognitive Neuroscience and Learning, Beijing Normal University, China).

6.2 Machine Learning Approaches

Multidimensional machine learning methods have only recently become in vogue in the analysis of fMRI data. In most studies, brain activity is averaged over space and time and frequently across participants to achieve proper signal-to-noise ratios [Haynes and Rees, 2005], but with the drawback of loosing information on spatial or temporal patterns.

Most fMRI studies measure voxel-wise correlation coefficients both for selecting informative voxels and for reporting the significance of the findings. This mass-univariate approach has several caveats as illustrated in a meta-analysis of 55 fMRI studies [Vul et al., 2009]. Based on reliability assumptions the highest expected correlation is claimed to be 0.74, which is surprisingly often exceeded in the surveyed studies. The *non-independence error* was identified to be the major reason for this discrepancy. Half of the surveyed studies used the same data for selecting a subset of voxels being correlated to the behaviour and reported the correlation on the very same data — no independent test data was used for verification. In machine learning terms, this corresponds to reporting the training error instead of the test error — the latter is regularly worse. VUL et al. propose two alternatives to avoid the non-independence error. Either, one should select voxels before examining behavioural data or one should split the data — one subset should be used for selecting voxels the second one for computing the actual correlation.

General Linear Models In classical fMRI analysis, the *general linear model* (GLM) [Friston et al., 1995] is the most frequently applied method and often acts as a baseline model for comparing alternative approaches. It models the activity in each voxel in a series of fMRI volumes by a linear combination of basis functions to derive a *statistical parametric map* (SPM). Such maps are suited to visualise task-specific differences in brain activity and for statistical inference.

Independent Component Analysis Traditionally, *independent component analysis* (ICA) has been applied in the field of blind source separation. Time-dependent fMRI data can be regarded as a complex mixture of high-frequent and low-frequent, task-specific and non-task specific activation patterns. ICA has successfully been used to extract such task-specific activation patterns [McKeown et al., 1998] and to extract functional cortical maps solely from their time-dependent activation [Bartels and Zeki, 2004]. Interestingly, a recent study has shown that ICA can effectively decompose neuroimaging data not due to its ability to select independent components but because it finds sparse components [Daubechies et al., 2009]. Consequently, the authors recommend to use algorithms that are in particular designed for sparsity to analyse fMRI data — the SFM is such a method.

Linear Discriminant Analysis Conventional fMRI scanners acquire brain volumes at a resolution of $3 \times 3 \times 3$ mm. Thus, it seems infeasible to characterise patterns below this spatial resolution such as the orientation-selective regions in the visual cortex V1. However, by combining univariate statistics and linear discriminant analysis the orientation of a visual stimulus can be predicted from such images [Haynes and Rees, 2005]. In this study, the most discriminative individual voxels were identified by applying voxel-wise t-tests. Assuming normal distributed classes with equal covariance matrices, a novel data point was assigned to the class with smallest Mahalanobis distance.

Support Vector-based Approaches The *searchlight* approach [Haynes et al., 2007] places spherical clusters — the searchlights — on each voxel, and the voxels within each sphere form a single data point. Then, standard linear support vector machines are used for decoding the brain state.

In [Davatzikos et al., 2005], the authors successfully discriminated emotional brain states using a whole-brain support vector classifier with Gaussian kernels even if the data was subsampled and averaged. The most discriminative regions were identified, however, no information was obtained about the minimum number of voxels that are necessary to discriminate two states.

Support vector machines for classifying whether a person has seen a fearful or a neutral face [Pessoa and Padmala, 2007] were found to perform significantly better if voxels from multiple brain regions were considered instead of voxels exclusively from a single region. The authors concluded that information is distributed across multiple regions, and multivariate approaches were proposed to exploit the distributed information in a *synergistical* way — better than any univariate approach could do. The support feature machine exactly provides such a synergistical way.

The major difference between the different support vector-based approaches is the way in which the dimensionality of the input data is reduced prior to training. Besides subsampling, other techniques such as explicit feature selection or *principal component analysis* (PCA) have been used successfully. In [Mourão Miranda et al., 2005], the authors used loss-less PCA, i.e. the input data was rotated such that the axes of the transformed coordinate system were parallel to the eigenvectors of the raw data. Afterwards, linear support vector machines were trained to discriminate brain activity data. The obtained weight vector may then be interpreted as a *discriminating volume* — large positive and negative values indicate high activity in the first and second task, respectively.

Alternative Imaging Methods Today, fMRI is still expensive, non-portable and inapplicable in natural environments. It cannot be integrated into a ubiquitous brain-computer interface. A more user-friendly interface could rely on *functional near-infrared spectroscopy* (fNIRS) to measure the cortical activity, however, with a much lower spatial resolution. In a recent study [Hosseini et al., 2011] based on fNIRS data, SVMs were used to predict whether a subject likes or dislikes a particular image. After reducing the dimensionality of the input data by PCA, a linear SVM achieved a prediction performance of 72.9% for attractive and 68.3% for unattractive stimuli. Here, the SVM captured the multidimensional dependencies, however, no cross-participant prediction is yet feasible due to the imaging method.

Affective States Only recently, the general feasibility of decoding affective states even across participants has been shown [Baucom et al., 2012]. In a preprocessing step, the number of voxels was reduced to the set of most stable voxels — i.e. those voxels that showed the most consistent variation across all stimuli [Mitchell et al., 2008]. A logistic regression classifier for discriminating valence and arousal levels performed significantly above chance. The authors conclude that information on valence and arousal are represented in *whole brain activation patterns*. The highest accuracy was obtained using 400 voxels ($3 \times 3 \times 3$ mm) for within-participant prediction and 2000 voxels for across-participant prediction.

There is strong evidence that affective information flows between communicating brains [Anders et al., 2011], i.e. emotion-specific information is encoded in a similar way in the sender's and the perceiver's brain but with temporal delay. Likewise, the brain activity between a speaker and a listener is coupled and the amount of coupling might be used as a measure for the success of the communication [Stephens et al., 2010].

6.3 Localised Brain Activity

In the following, we demonstrate how the repetitive SFM (RSFM, see Chapter 3.2) can be used to effectively identify informative voxels in an FMRI dataset recorded during a simple motor task [KLEMENT et al., 2013]. Participants were asked to press a button in their left or right hand. The goal was to identify voxels that discriminate between left vs. right button presses. We will see that the RSFM finds the relevant voxels (as identified with standard univariate approaches) with high accuracy and might be used to derive an estimate of the total number of informative voxels.

Data acquisition We used FMRI data that was acquired from 12 healthy female participants (mean age 21.6 years, range 19 to 26 years) in a 3 Tesla scanner (Philips Medical Systems). Sixty-seven functional whole-brain images were acquired during each of a total of four runs per participant (T_2^* weighted echo-planar images, 42 horizontal interleaved slices, tilt angle -30°, 3 mm slice thickness, in plane resolution 3 × 3 mm², FOV 240 × 240 mm², TE 35 ms, TR 3000 ms). Participants were shown short text messages (either *happy* or *sad*) through FMRI-compatible video goggles and asked to decide whether they wanted to press a button in their left or right hand immediately whenever a text message appeared on the screen, but to hold their decision in mind and to execute their decision only when a go-signal (two arrow heads, one pointing to the left and one pointing to the right) appeared on the screen. Participants were instructed to respond as quickly as possible when the go-signal appeared by pressing the selected button with their left or right thumb, respectively. During each run, 12 trials (mean duration 5 scans) were presented in pseudo-randomised order, using the following timing parameters: stimulus presentation time 1000 ms; delay 2000 or 3500 ms; go-signal 300 ms; inter-trial interval 8700 to 13200 ms (steps of 1500 ms). The study was approved by the Ethics Committee of the University of Lübeck.

Preprocessing The preprocessing included removal of the first two functional scans of each run, slice acquisition time correction, concurrent spatial realignment and correction of image distortions, normalisation into standard MNI space (Montreal Neurological Institute), and spatial smoothing with an 8 mm FWHM (full width half maximum) Gaussian kernel. Individual activity maps for left-hand and right-hand button presses were estimated for each participant and run using a standard GLM procedure. In short, the amplitude of each participant's voxel-wise brain activity was estimated with an individual linear model that contained separate regressors for the predicted time course of BOLD activity associated with left and right button presses (stick functions convolved with a canonical hemodynamic response function as provided with SPM5). Additional regressors were included in these models to account for low-frequency drifts (cut-off period 128 s) and BOLD activity in response to text messages. High-frequency artefacts were

accounted for by removing first-order autocorrelations. This procedure revealed eight activation maps for each participant (four for left-hand button presses and four for right-hand button presses). The overall dataset — in the following referred to as the *buttonpress* dataset — consists of 96 brain volumes (2 conditions × 12 participants × 4 runs) each of which contains 50989 in-brain voxels identified with the brain mask published by [Tzourio-Mazoyer et al., 2002].

Mass Univariate Analysis The traditional approach to identify voxels in fMRI data that show different levels of activity during two conditions (i.e. that are discriminative) are voxel-wise univariate analyses [Holmes et al., 1997]. Here, we used such a mass univariate approach to assess the overlap between voxels identified as being discriminative with the univariate approach and those identified as being relevant with SFM-based repetitive feature selection. For this, activation maps of each participant were averaged for each class and fed into a group-level voxel-wise paired t-test, and the overlap between voxels with high absolute t-values and voxels identified as being relevant with the RSFM was assessed. This approach is very similar to Golub's correlation-based feature selection (see Chapter 2.6.3). For two equally-sized classes,

Anatomical region name	Significant voxels left	right
Postcentral gyrus	25.0% (319)	16.0% (204)
Precentral gyrus	7.4% (94)	7.5% (96)
Cerebellum VI	4.1% (52)	6.1% (78)
Inferior parietal lobe	5.3% (67)	1.3% (16)
Cerebellum IV/V	2.5% (32)	3.1% (39)
Putamen	1.5% (19)	0.0% (0)
Supplementary motor area	1.3% (17)	0.0% (0)
Superior parietal lobe	1.3% (16)	1.3% (16)
Pallidum	1.0% (13)	0.0% (0)
Supramarginal gyrus	0.9% (11)	0.4% (5)
Unassigned	9.0% (115)	
Other regions (< 1%)	5.1% (65)	

Figure 6.1: Discriminative voxels as identified by the mass univariate approach (2.5% most significant voxels, voxel-wise t-statistics with $p \leq 0.001$). Anatomical regions were identified by an automatic labelling procedure [Tzourio-Mazoyer et al., 2002, Schmahmann et al., 1998]. Only regions that contain at least 1% of all significant voxels across hemispheres are listed. The number of discriminative voxels in each region are shown in brackets.

Golub's correlation coefficient and Student's t-value are equivalent. For comparison with the sfm, the top 2.5% of all voxels (n = 1274, voxel-wise $p \leq 0.001$) were deemed discriminative (see Figure 6.4; note, in this figure the results of the mass univariate method are compared to those later obtained by the sfm). An automatic labelling procedure based on the anatomical parcellation of the mni single-subject brain [Tzourio-Mazoyer et al., 2002, Schmahmann et al., 1998] (see Figure 6.1) confirmed that three major anatomical regions — the *precentral gyrus (primary motor cortex)*, the *postcentral gyrus (primary somatosensory cortex)* and a region in the *cerebellum* — contain discriminative voxels.

Support Vector Machine with Random Feature Selection To obtain a baseline estimate of classification performance, we trained a linear hard-margin svm on randomly chosen d-dimensional feature subsets (d = 1, 2, 4, …, 32768) with 44 samples in each class (11 participants × 4 runs per class) in a leave-one-participant-out cross-validation scheme. This procedure was repeated 1000 times for each subset size. As can be seen in Figure 6.2, the test error was below chance for the large majority of repetitions even if only a single dimension was selected at random, and close to zero if more than 1000 features were included, indicating a strong degree of redundancy in the data.

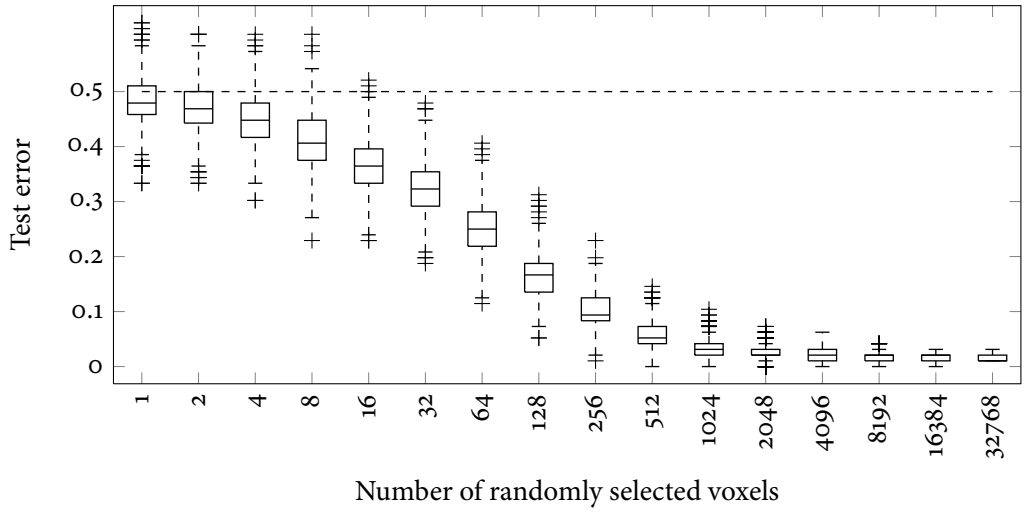


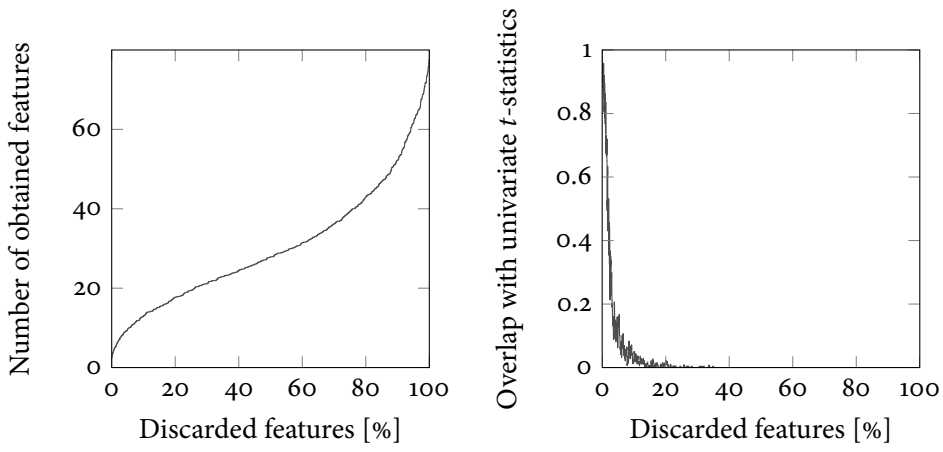
Figure 6.2: Randomised support vector approach. The boxplot visualises the distribution of the leave-one-participant-out cross-validation error (median, lower and upper quartile, outliers). For a sample size of 32, the error is below chance in every single run.

Results with the Repetitive Support Feature Machine To reduce runtime, the data was down-sampled for use with the SFM. Originally, the data contains 50989 dimensions. We chose a down-sampling factor of two in each direction, which reduced the data by a factor of eight (precisely, the number of voxels was reduced to 6362 due to the irregular shape of the brain). Given that the data was spatially smoothed with an isotropic kernel of 8 mm FWHM (full width at half maximum) during preprocessing, this should not reduce the information content of the data. Using the Gaussian pyramid with an optimised boundary condition during filtering (see Chapter 5) the number of voxels may be reduced even further (see Chapter 6.4.4).

To estimate the fraction of informative and uninformative voxels in this dataset, we plotted (i) the size of the feature subset, (ii) the test error of the RSFM and (iii) the test error of an SVM trained on the features that remained in the dataset after all features identified by the SFM in a particular repetition had been removed (see Figure 6.3). In this particular dataset, we used a soft SVM to test whether the remaining features after each run still contained information relevant for classification. The optimal softness of this SVM was estimated by a similar leave-one-participant-out cross-validation scheme as used for the RSFM. Thus, for each set of remaining features computed on $2 \times 11 \times 4 = 44$ vs. 44 samples we trained the SVM on $2 \times 10 \times 4 = 40$ vs. 40 samples and tested on the 11th subject. Once the optimal softness parameter was determined, the SVM was retrained on all 11 subjects and tested on the 12th subject. This way, a function representing the test error of an optimised soft SVM over repetitions was derived for each of the 12 participants.

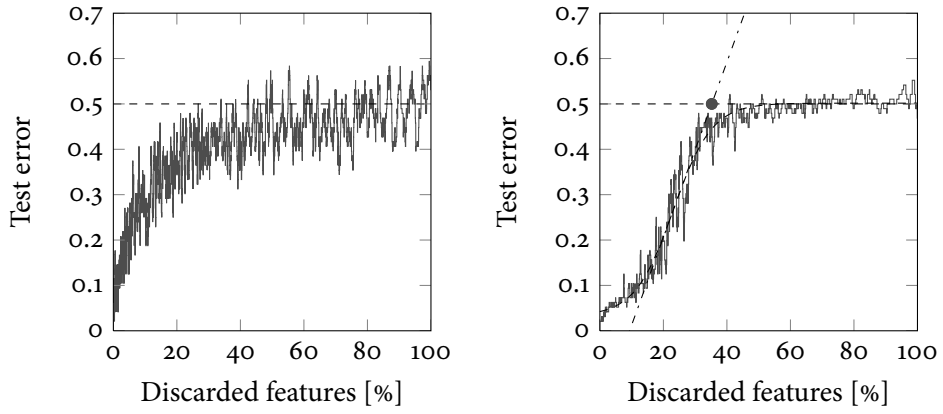
Since we used a leave-one-participant-out scheme for cross-validation, the feature set size and error functions obtained during each validation do not necessarily have the same length (i.e. the number of repetitions until all features are consumed may differ across validation runs). Thus, these functions need to be re-sampled before averaging. We chose a re-sampling procedure in which feature subsets were first sorted according to their size and each $x \in 1, \ldots, d$ was then assigned with the performance value of the last repetition in which less than x features were removed. These piece-wise constant curves were then averaged across all leave-one-participant-out cross-validations.

A comparison of voxels obtained by the RSFM and voxels identified as being discriminative by univariate t-statistics revealed a large overlap (see Figure 6.3, top right and Figure 6.4). This indicates that the RSFM very quickly consumes significant features before other features are included. The smallest feature subset contained 2.4 voxels on average. The largest feature subset contained an average of 77.7 voxels, which is below the upper bound (i.e. number of data points ($n = 88$ samples) minus 1 (VC-dimension of a linear classifier), see Figure 6.3, top left). The test error of both the RSFM and the SVM converged to chance level as more and more features were discarded (see Figure 6.3, bottom). However, due to large repetition-to-repetition fluctuations



(a) Relevant features (single-run SFM)

(b) Overlap relevant/significant features



(c) Test error (single-run SFM)

(d) Test error (SVM on remaining features)

Figure 6.3: Analysis of the *buttonpress* dataset with the RSFM and an SVM trained on the remaining features. Shown are the average number of relevant features (a), the average overlap between the features identified with the repetitive SFM and those that were found to be discriminative with voxel-wise t-statistics (2.5% most significant features, $p \leq 0.001$) (b), the average leave-one-participant-out cross-validation error of the SFM (c), and the average leave-one-participant-out cross-validation error of an SVM trained on the remaining features (d). To approximate the number of features (voxels) that carry information, a sigmoid function (dashed) was fitted to the test error function of the SVM. The straight line (dash-dotted) through the inflexion point of the sigmoid crosses chance level at 35% (•), indicating that no more than 35% of all voxels in this dataset carry movement-specific information.

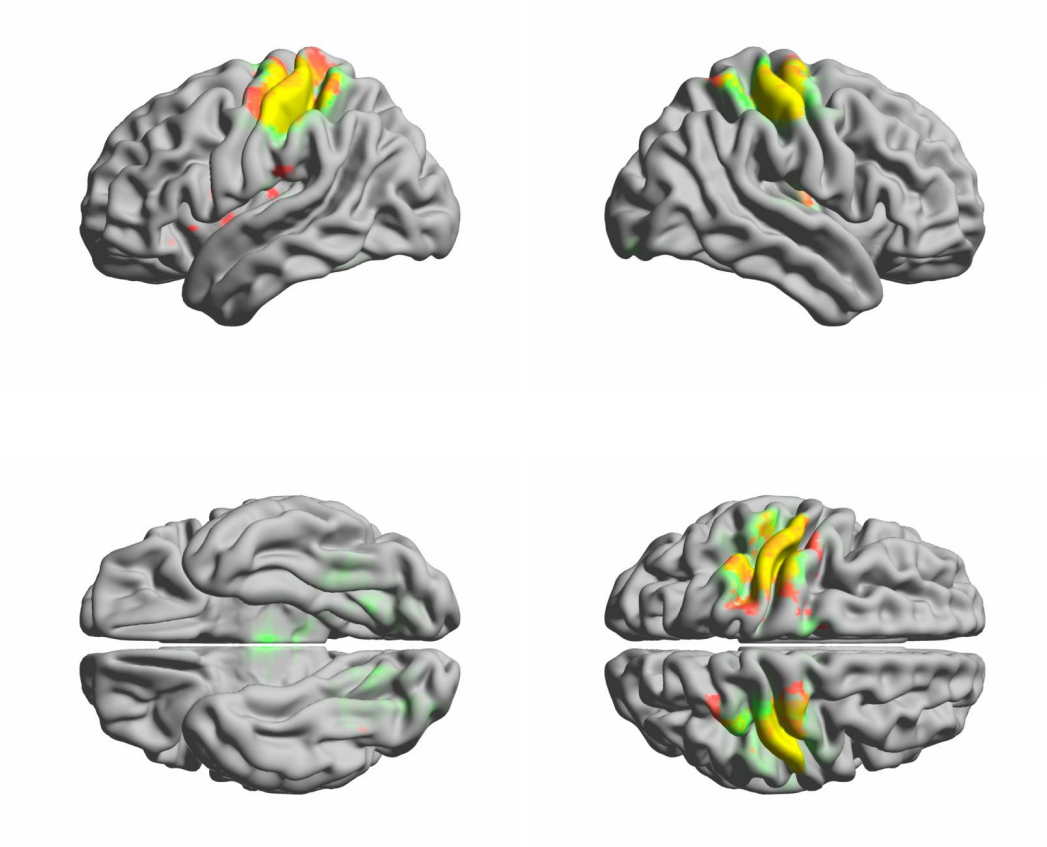
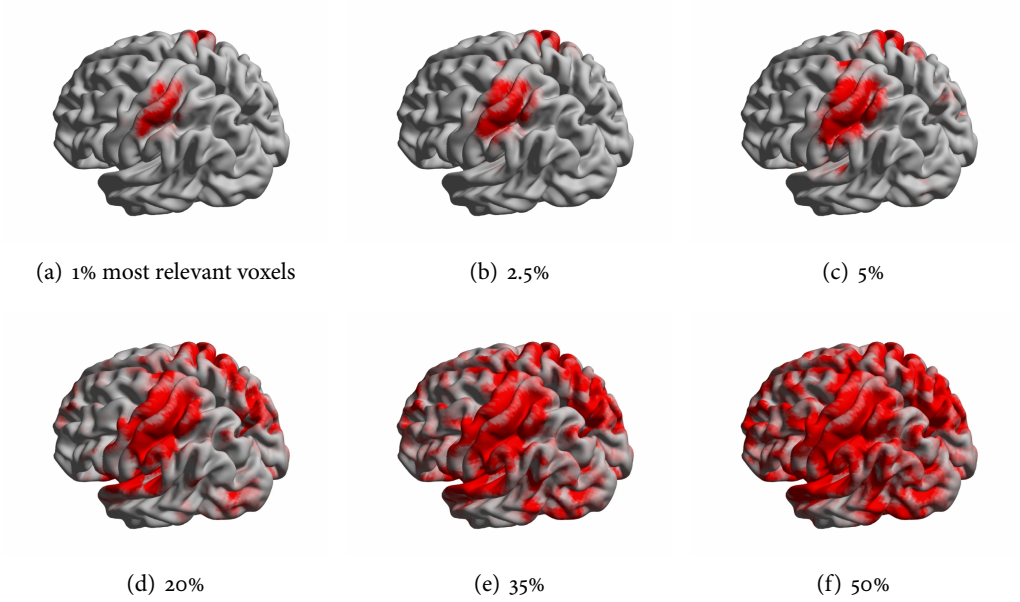


Figure 6.4: Voxels found to be relevant for discriminating the two tasks in the *buttonpress* experiment. Discriminative voxels as identified by voxel-wise t-statistics (2.5% most significant voxels, $p \leq 0.001$) are red, voxels (features) found to be relevant by the RSFM (cut-off 2.5% of all voxels) are green. Overlapping regions are yellow. Colour intensity indicates depth below surface, i.e. bright red regions are close to the surface, while faint red regions are located deeper in the brain. Additionally, for the RSFM, colour intensity indicates how consistent a specific voxel was chosen across participants, i.e. bright green regions were consistently identified for all left-out-participants, while faint green regions were only identified in few cross-validation runs. Discriminative voxels are mainly located in the *precentral* and the *postcentral gyri* (motor and somatosensory cortex), with a high degree of overlap between the two methods. The cerebellum is not shown.

of the test error, the point where the test error is no longer different from chance is difficult to derive, particularly for the RSFM. To approximate that point, we fitted the sigmoid function

$$f(x) = \alpha_0 + \frac{\alpha_1}{1 + e^{-\frac{x-\alpha_2}{\alpha_3}}}$$

to the test error function of the SVM. The coefficients α_0 to α_3 were estimated using least-squares approximation. The point at which the remaining features contained no more information was defined as the intersection point of a straight line through the inflexion point with the same slope as the sigmoid at that point and chance level (Figure 6.5, bottom right). In the *buttonpress* dataset this point was reached when approximately 35% of all voxels were discarded. Figure 6.5 shows how the distribution of voxels identified by the RSFM as being relevant evolves over repetitions. Given our estimate that about 35% of all voxels carry information relevant for classification, the second last plot in the second row marks that point where all (even weakly) informative voxels



(a) 1% most relevant voxels (b) 2.5% (c) 5%

(d) 20% (e) 35% (f) 50%

Figure 6.5: Location of the most relevant voxels in the *buttonpress* dataset as obtained by the SFM. Shown are those voxels that were consistently selected within the 12 cross-validation runs as the most relevant voxels, i.e. they were among the first 1% of the selected features (a), 2.5% (b) and so on. Obviously, the brain activity remains localised in two clusters (a–d) up to a certain amount of features. Then, more and more weakly relevant features from all over the brain are obtained.

have been identified (marked in red). As can be seen, these voxels were mainly located in two dense clusters in the motor and somatosensory cortex of both hemispheres. Pushing the RSFM close to and beyond this limit returned voxels that were more or less scattered across the whole brain because more and more weakly informative or non-informative features were included. This provides additional evidence that our estimate of the fraction of relevant voxels is a valid approximation of the amount of truly informative voxels.

In sum, our results show that the proposed repetitive application of the SFM identifies informative features very effectively even in datasets that contain several informative feature subsets that all permit linear separation such as the FMRI dataset used here. Analysing the test error function of an SVM trained on the features discarded by the RSFM in each run revealed an estimate of the fraction of informative voxels that converges with neuroscientific considerations. It is important to note that the accuracy of this estimate relies heavily on the *selectivity* of the applied feature selection method. If the method used for feature selection falsely identifies irrelevant (uninformative) features as relevant, then the proportion of relevant features in the dataset is overestimated. Since the SFM is very restrictive in the way it selects relevant features and returns a high percentage of truly relevant features (see Chapters 4.2 and 3.3), the estimate of informative features obtained with the RSFM likely represents an unbiased estimate. As we use a repetitive approach, the estimate of the proportion of informative features does not strongly rely on the *sensitivity* of the feature selection — even if only one (truly relevant) feature is obtained in each repetition, the proportion of informative voxels will still be estimated without bias.

6.4 Emotional Brain States

The *buttonpress* dataset can effectively be decoded with mass univariate methods; the SFM-based approach is equally well suited. But, the SFM is an intrinsically multidimensional method and may therefore be used for complex activity patterns such as in the following study.

Data Acquisition Affective brain states were analysed based on FMRI data from six healthy female participants (mean age 22 years, range 20 to 25 years). Participants gave their written informed consent prior to participation and the study was approved by the ethics committee of the University of Tübingen.

Each participant was informed that her task would be to submerge herself into emotional situations and to facially express her emotional feelings as soon as they arise. To maximise the participants compliance, they were told that their romantic partner would watch them and that he would have the task to share her feelings. Scanning consisted of ten runs; each run comprised four 20 s trials during which affective information was to be expressed, and five periods during

which the participant was instructed to relax (24 s, 22 s, 18 s, 22 s, 18 s). A single emotion (*joy*, *anger*, *disgust*, *fear*, or *sadness*) was used in each run in order to avoid rapid switches between conflicting emotions. A single printed word (e.g. *joy*) signalled emotion periods to the participant. The order of emotions was chosen by the participant, with the restriction that no emotion could occur twice in a row and that each emotion had to be chosen once before an emotion could be chosen a second time. Ninety-two functional images covering the whole brain were acquired during each run (T_2^* weighted echo-planar images, 1.5 Tesla Siemens Avanto, Erlangen, Germany; tilt angle -30°, 64 × 64 matrix, in plane resolution 3 × 3 mm, 24 axial slices, interleaved order, slice thickness 6 mm with no gap, TE 40 ms, TR 2000 ms).

Preprocessing Preprocessing included slice acquisition time correction, concurrent spatial realignment and correction of image distortions by use of individual static field maps, normalisation into standard MNI space and spatial smoothing (10 mm Gaussian kernel). For each 20 s trial, a single beta image was estimated using a GLM. The final dataset contained 240 datapoints (6 participants × 8 trials × 5 emotions) with 41642 voxels.

6.4.1 Pairwise Emotion Analysis

First, we focus on discriminating pairs of emotions, before we proceed with classifying one emotion against all others. In all settings, prediction performance across participants was evaluated by training on all but one participant and testing on the remaining — i.e. 6 cross-validation runs, 80 data points for training (5 participants × 8 trials × 2 classes), 16 data points for testing (1 participant × 8 trials × 2 classes).

Randomised Support Vector Machine As for the *buttonpress* dataset we trained linear support vector machines on randomly selected d-dimensional voxel subsets (see Figure 6.6). We observed that the pair *joy* vs. *sadness* was best separable by a linear SVM, while the pairs *fear* vs. *sadness* and *anger* vs. *sadness* were almost inseparable. Although emotions are supposed to involve multiple regions, in some cases even a single feature may be suited for separation — the average cross-validation error was less than chance. This randomised approach gives a hint to which emotions are well separable and which are not.

First Iteration of the SFM The very first iteration of an SFM on the *emotion* dataset revealed that on average 2.3 to 8.0 features were required to separate the training data (see Figure 6.7, b). The average test error ranged from 0.16 (*joy* vs. *anger*) to 0.42 (*fear* vs. *sadness*). The emotion *joy* was well separable from all other emotions — except for *disgust*. To assess incidental separability, we randomly permuted the class labels for each participant separately and repeated

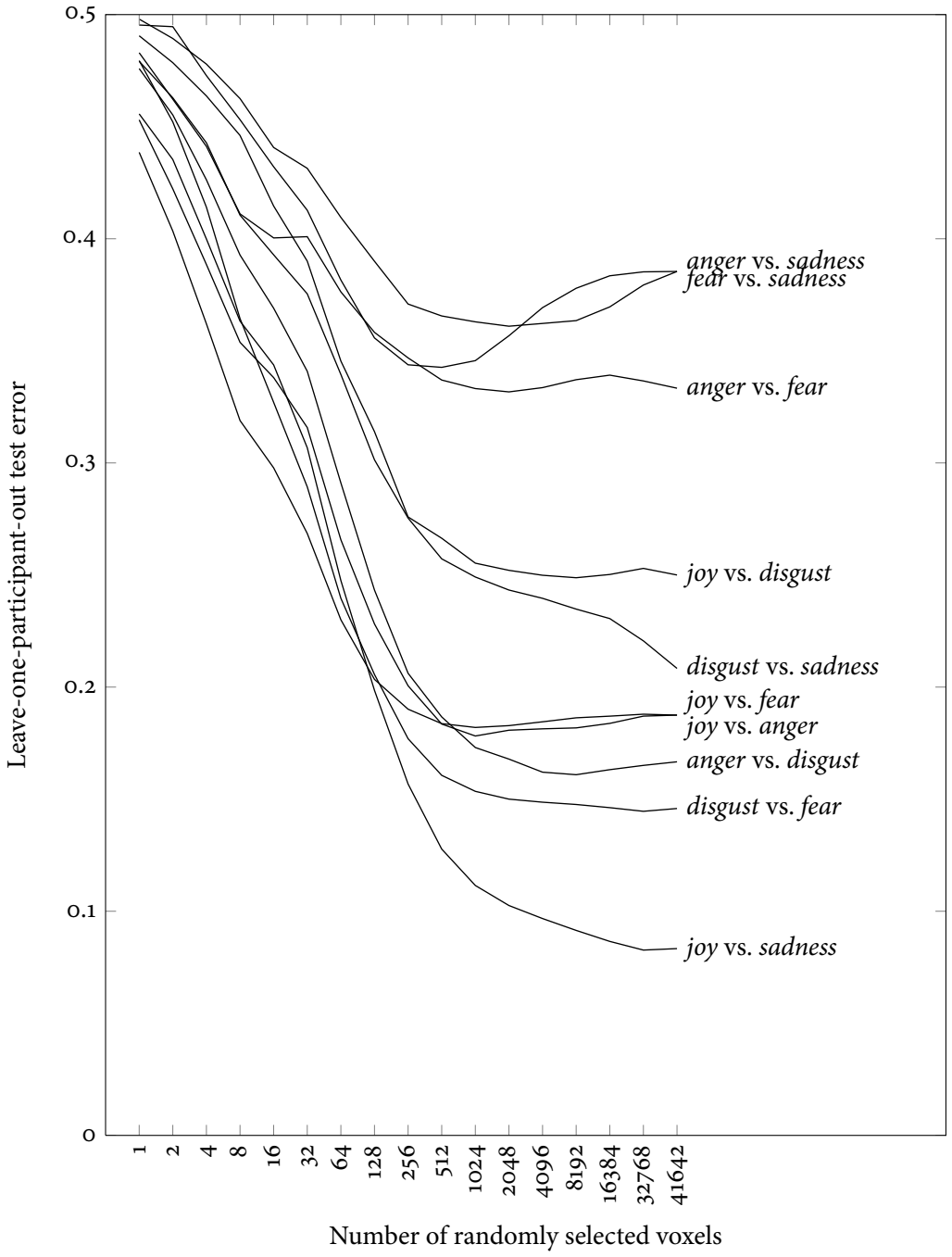


Figure 6.6: Randomised support vector approach for pairwise emotion analysis. Shown are the mean leave-one-participant-out cross-validation errors for all 10 emotion pairs (averaged over 1000 repetitions).

the experiment (see Figure 6.8). Indeed, chance level was 0.5, while the number of relevant features ranged from 15.5 to 16.3. This is far less than the theoretical upper bound of 79 which is the VC-dimension of a linear classifier trained on 80 data points in a high-dimensional space. Obviously, the large number of dimensions causes the solution space to contain solutions with comparatively few features. Furthermore, although the test error of *fear* vs. *sadness* is very poor (0.42), the number of relevant features (7.5) is far below chance level. In comparison to the randomised SVM approach, the SFM produces an error only slightly worse than the error produced by an SVM on the complete feature set. For *joy* vs. *anger*, the SFM identifies 2.3 features on average with a test error of 16%, while an SVM trained on the overall feature set produces only a slightly larger test error of 18.8%.

(a) Number of obtained features

	2nd Emotion				
1st Emotion	Joy	Anger	Disgust	Fear	Sadness
Joy	—	2.3 (± 0.5)	4.8 (± 0.7)	3.5 (± 1.1)	4.0 (± 0.6)
Anger	2.3 (± 0.5)	—	5.3 (± 0.9)	7.5 (± 0.8)	6.8 (± 1.1)
Disgust	4.8 (± 0.7)	5.3 (± 0.9)	—	4.7 (± 0.7)	8.0 (± 2.2)
Fear	3.5 (± 1.1)	7.5 (± 0.8)	4.7 (± 0.7)	—	7.5 (± 0.5)
Sadness	4.0 (± 0.6)	6.8 (± 1.1)	8.0 (± 2.2)	7.5 (± 0.5)	—

(b) Cross-validation error

	2nd Emotion				
1st Emotion	Joy	Anger	Disgust	Fear	Sadness
Joy	—	0.16 (± 0.15)	0.33 (± 0.12)	0.19 (± 0.18)	0.23 (± 0.14)
Anger	0.16 (± 0.15)	—	0.25 (± 0.16)	0.24 (± 0.14)	0.34 (± 0.13)
Disgust	0.33 (± 0.12)	0.25 (± 0.16)	—	0.28 (± 0.12)	0.32 (± 0.19)
Fear	0.19 (± 0.18)	0.24 (± 0.14)	0.28 (± 0.12)	—	0.42 (± 0.19)
Sadness	0.23 (± 0.14)	0.34 (± 0.13)	0.32 (± 0.19)	0.42 (± 0.19)	—

Figure 6.7: Leave-one-participant-out cross-validation results for each emotion pair using the SFM. Shown are the number of obtained features (a) and the error rate (± standard deviation) (b). Obviously, *joy* can be distinguished very much better from other emotions than *sadness* or *anger*, e.g. *joy* is separable from *sadness* with an error of 23% using only 4.0 voxels, while *fear* and *sadness* are separable with an error of 42% using 7.5 features on average.

(a) Number of obtained features

1st Emotion \ 2nd Emotion	Joy	Anger	Disgust	Fear	Sadness
Joy	—	15.6 (± 2.5)	16.1 (± 2.7)	16.3 (± 2.4)	15.5 (± 2.6)
Anger	15.6 (± 2.5)	—	15.7 (± 2.4)	15.5 (± 2.2)	16.0 (± 2.4)
Disgust	16.1 (± 2.7)	15.7 (± 2.4)	—	15.8 (± 2.7)	15.9 (± 2.1)
Fear	16.3 (± 2.4)	15.5 (± 2.2)	15.8 (± 2.7)	—	16.0 (± 2.3)
Sadness	15.5 (± 2.6)	16.0 (± 2.4)	15.9 (± 2.1)	16.0 (± 2.3)	—

(b) Cross-validation error

1st Emotion \ 2nd Emotion	Joy	Anger	Disgust	Fear	Sadness
Joy	—	0.50 (± 0.11)	0.50 (± 0.11)	0.52 (± 0.10)	0.50 (± 0.11)
Anger	0.50 (± 0.11)	—	0.49 (± 0.11)	0.48 (± 0.10)	0.48 (± 0.11)
Disgust	0.50 (± 0.11)	0.49 (± 0.11)	—	0.50 (± 0.11)	0.48 (± 0.10)
Fear	0.52 (± 0.10)	0.48 (± 0.10)	0.50 (± 0.11)	—	0.49 (± 0.10)
Sadness	0.50 (± 0.11)	0.48 (± 0.11)	0.48 (± 0.10)	0.49 (± 0.10)	—

Figure 6.8: Leave-one-participant-out cross-validation results for each emotion pair using the SFM after randomly permuting emotion labels within each participant. Shown are the averaged results of 20 random permutations. As expected, we observe an average error rate of 50%, but the number of features remains low (15.5 – 16.3) compared to the theoretical upper bound of 79 (VC-dimension of a linear classifier trained on 80 data points). Thus, even for random data the obtained feature set may be small.

Repetitive Support Feature Machine A single run of the SFM quantifies how well the classes are separable with the least number of features. However, it does not capture the amount of redundancy that we expect to be large for affective states. Additionally, as the SFM may fail to converge, the true number of relevant features may even be less than in the very first iteration. Therefore, we evaluated the repetitive support feature machine in the same leave-one-participant-out cross-validation fashion as in the *buttonpress* experiment. We compared a well separable pair (*joy* vs. *sadness*) and a second almost inseparable pair (*fear* vs. *sadness*) (see Figure 6.9). The amount of redundancy was again evaluated by training SVMs on the remaining features, but we used the median — instead of the mean — to combine the six leave-one-participant-out runs, because the median is less affected by outliers. As in the *buttonpress*

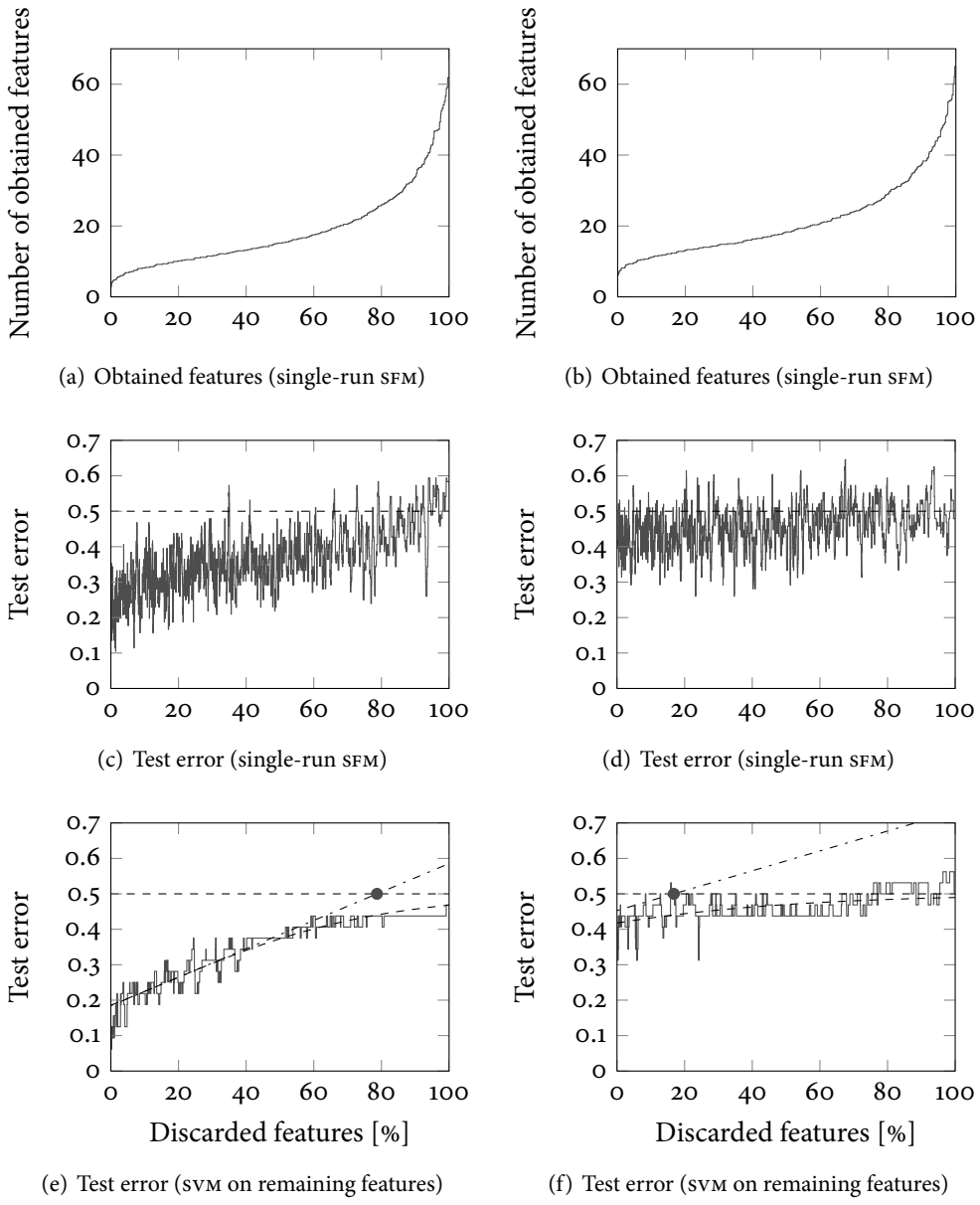
experiment, a sigmoid was fitted to the obtained error values, however, the sigmoid was less prominent than in the *buttonpress* experiment and a least-squares fit gave numerically unstable results. Thus, we further restricted the sigmoid to the domain $]0, 0.5[$, i.e. we fitted

$$f(x) = \frac{0.5}{1 + e^{-\frac{x-\alpha_2}{\alpha_3}}} .$$

For the well separable pair *joy* vs. *sadness* the SFM effectively filters out relevant feature sets (see Figure 6.9). Almost all feature sets produce a test error less than chance. Surprisingly, even if the 80% most relevant features were discarded, the SVM still performs better than chance. Thus, we conclude that — neglecting inaccuracies of the SFM – almost the whole brain carries discriminative information. For the almost inseparable pair — *fear* vs. *sadness* — the behaviour is completely different. Only very few feature sets are relevant and the performance of an SVM on the remaining features quickly converges to 0.5. For both pairs we derived an estimate of the information distribution via the straight line through the inflexion point of the fitted sigmoid. Note, that for *fear* vs. *sadness* this inflexion point is not within the visible range. Using this estimate, we found 79% of all features to be informative for *joy* vs. *sadness* while only 17% carried information for *fear* vs. *sadness*.

Finally, we visually compared the regions that were found to be relevant in both pairs (see Figures 6.10 and 6.12). The discriminating regions seem to only slightly overlap. We further analysed this overlap across all ten emotion pairs, i.e. we identified the brain regions that were most frequently found to be relevant. For each of the ten emotion pairs, we selected those voxels that were within the 20% most relevant voxels in at least four of the six cross-validation runs (more than half). We then combined the results across emotion pairs by keeping only those voxels that were present in at least half of all emotion pairs. These voxels have a large probability to be relevant for all emotion pairs across all participants. An automatic anatomical labelling of these voxels (see Figure 6.11) reveals that the largest overlap is located in the orbital part of the *inferior frontal gyrus* (Brodmann area 47). This region was shown to be involved in computing the relevance of emotional information [Beer et al., 2006] especially reward and punishment [Kringelbach and Rolls, 2004]. Thus, our findings are consistent with classical neuroscientific research.

These results indicate that pairwise SFM-based emotion analysis effectively extracts relevant regions to discriminate affective brain states. *Joy* is the emotion that can be best discriminated from all other emotions. Further, affective information seems to be highly distributed across the whole brain — we found up to 80% of the brain voxels to carry discriminative information.



(a) Obtained features (single-run SFM)

(b) Obtained features (single-run SFM)

(c) Test error (single-run SFM)

(d) Test error (single-run SFM)

(e) Test error (SVM on remaining features)

(f) Test error (SVM on remaining features)

Figure 6.9: Analysis of the *emotion* dataset with the repetitive SFM. Shown are the number of obtained features (a,b), the leave-one-participant-out cross-validation error of the SFM (c,d), and the median of the leave-one-participant-out cross-validation error of an SVM trained on the remaining features (e,f) for a well separable emotion pair (*joy* vs. *sadness*, left column) and an almost inseparable emotion pair (*fear* vs. *sadness*, right column). The numbers of voxels that carry information were again approximated by fitting sigmoids.

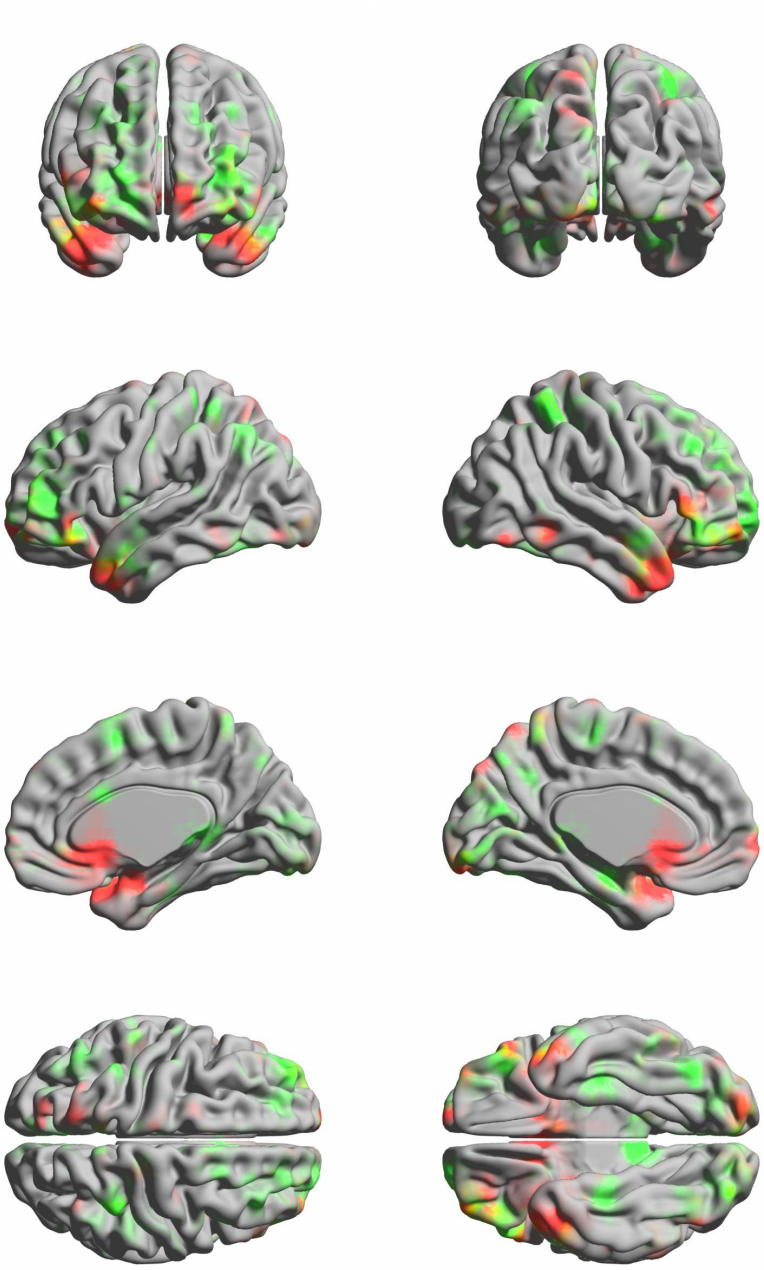


Figure 6.10: Regions found to be relevant to discriminate *joy* vs. *sadness* (red) and *fear* vs. *sadness* (green). Overlapping regions are coloured in yellow. For both emotion pairs the 2.5% most relevant features were included.

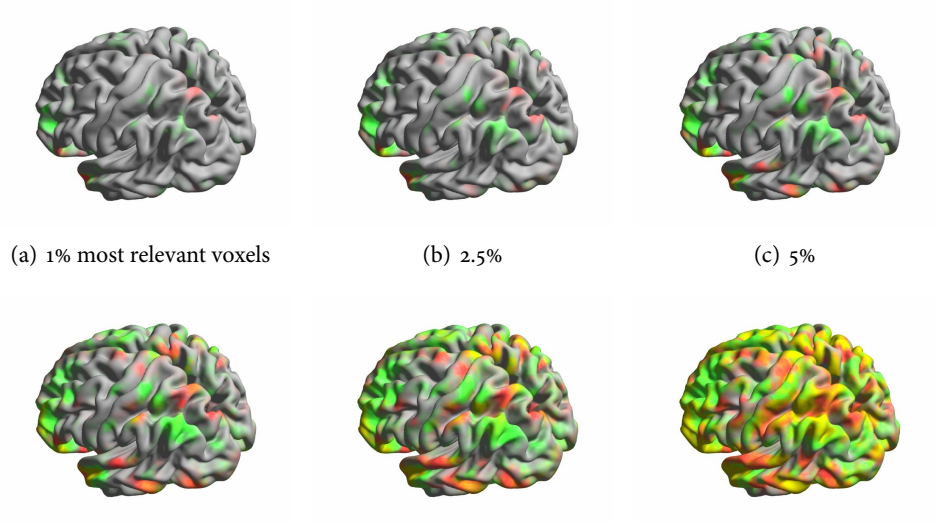
Anatomical region name	Significant voxels left		right	
Inferior frontal gyrus, orbital part	2.2%	(9)	6.0%	(24)
Middle frontal gyrus	3.2%	(13)	5.5%	(22)
Superior frontal gyrus	5.0%	(20)	4.5%	(18)
Superior frontal gyrus, medial	4.5%	(18)	3.0%	(12)
Inferior temporal gyrus	1.0%	(4)	4.5%	(18)
Parahippocampal gyrus	1.2%	(5)	3.7%	(15)
Middle temporal gyrus	2.5%	(10)	3.5%	(14)
Temporal pole: superior temporal gyrus	3.5%	(14)	3.5%	(14)
Middle frontal gyrus, orbital part	3.2%	(13)	2.7%	(11)
Superior frontal gyrus, medial orbital	3.0%	(12)	2.7%	(11)
Other regions (< 4%)	30.9%	(124)		
Total	100.00%	(401)		

Figure 6.11: Hemisphere-specific anatomical distribution of relevant voxels in the *emotion* dataset. Shown are those voxels that were consistently chosen to be relevant across all ten emotion pairs and participants (20% most relevant voxels *and* relevant in at least 4 out of 6 cross-validation runs *and* relevant in at least 6 out of 10 emotion pairs).

6.4.2 One-vs.-All Emotions Analysis

The next more complex question is whether a single emotion may be separated from all other emotions. To address this question, we combined data from all but one emotion by averaging across all 4 trials of one run such that we obtain 2 data points for each sender and each emotion. In this way we rebalance the input data to avoid artefacts due to unequal class sizes. We conducted the same leave-one-participant-out cross-validation analysis as for pairwise emotions — each training set contained 40 vs. 40 samples (8 trials × 5 participants vs. 2 average trials × 4 emotions × 5 participants) and the test sets contain 8 vs. 8 samples (1 emotion × 8 trials from one participant vs. 2 average trials × 4 emotions from the same participant).

The results of a single SFM run (see Figure 6.13) show that *joy* is the best separable single emotion with an average number of relevant features of 5.5 and a mean test error of 19%. All other emotions are separable with more features and less accuracy. A repetitive SFM reveals further differences in the ability to separate emotions (see Figure 6.14). The information on how to separate *joy* from all other emotions seems to be widely distributed and even the least relevant voxels still carry sufficient information to permit classification with an accuracy of 40%. In con-



(a) 1% most relevant voxels (b) 2.5% (c) 5%

(d) 10% (e) 20% (f) 40%

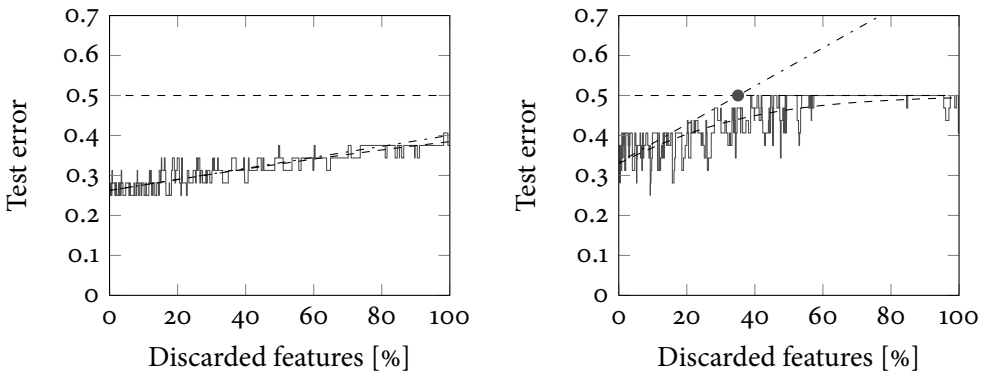
Figure 6.12: Location of relevant voxels for discriminating *joy* vs. *sadness* (red) and *fear* vs. *sadness* (green) as obtained by the SFM. Shown are those voxels that were consistently selected within the six cross-validation runs as the most relevant voxels, i.e. they were among the first 1% of the selected features (a), 2.5% (b) and so on. In contrast to the *buttonpress* dataset, relevant regions are spread all over the brain as small clusters. Additionally, the overlap between the two tasks (yellow) remains small throughout many repetitions of the SFM.

trast, information on how to separate *fear* from all other emotions is less prominent — accuracy reaches chance level when approximately 40% of all voxels were discarded.

These results further emphasise that affective information is distributed across large brain regions. However, some emotions — especially *joy* — are better separable and seem to involve larger regions while others cover a smaller proportion of the brain. Compared to pairwise emotion analysis the error rates and the obtained feature sets are larger. The classifiers were less accurate than in the pairwise scenarios. This may be due to the way in which data points were averaged: Balanced scenarios were obtained by averaging data of 4 out of 5 emotions — one emotion was included unchanged. Further, high-dimensional data is almost never located close to its mean, but the average of data points comes closer to the mean. Here, data from one emotion (raw data, far away from the mean) and data from all other emotions (averaged data, closer to the mean) are combined. So, rebalancing the data might have introduced another imbalance which alters the high-dimensional distribution.

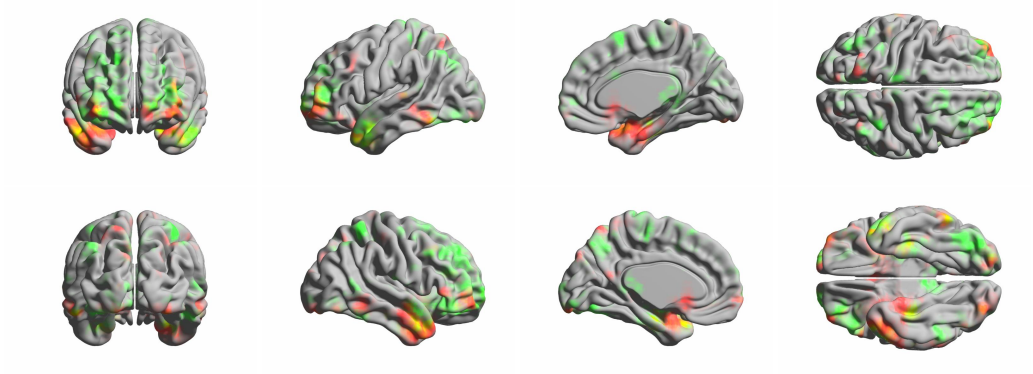
Emotion	Joy	Anger	Disgust	Fear	Sadness
Features	5.5 (± 1.0)	7.2 (± 1.0)	6.8 (± 1.5)	8.2 (± 2.6)	9.7 (± 1.6)
Error	0.19 (± 0.11)	0.34 (± 0.09)	0.29 (± 0.17)	0.35 (± 0.11)	0.46 (± 0.09)

Figure 6.13: Performance of the SFM in a one-vs.-all emotions scenario. Shown are the number of obtained features and the leave-one-participant-out cross-validation error (± standard deviation).



(a) Test error (SVM on remaining features)

(b) Test error (SVM on remaining features)



(c) Relevant regions

Figure 6.14: Repetitive SFM for classifying emotional brain states. Shown are the accuracy of an SVM on the remaining features for *joy* vs. all (a) and *fear* vs. all (b) and the relevant regions for both emotions (*joy* — red, *fear* — green, both — yellow).

6.4.3 Time Slice Analysis

In the previous evaluations, the input data consisted of beta images that were estimates of brain activity across a whole trial in which participants had to experience and express emotions. Thus, each feature is an approximation of the average activity change within a period of 20 s. However, such beta images do not allow to assess any time-dependent aspects of brain activity.

Preprocessing We slightly changed the preprocessing to get beta images at a higher time resolution. For each trial, we estimated one pre-emotional (-4 to 0 s), five intra-emotional (0 to 20 s) and three post-emotional (20 to 32 s) beta images each covering an interval of 4 s. Again, the average BOLD activity change was estimated using the hemodynamic response function (HRF) as provided by SPM5. All voxels that were undefined in at least one trial were discarded. Thus, we obtained 41180 voxels. Each time slice was labelled according to the timestamp (pre-emotional: -2 s; intra-emotional: 2 s, 6 s, 10 s, 14 s, 18 s; post-emotional: 22 s, 26 s, 30 s). Then, we conducted the same pairwise SFM-based information content analysis as before, i.e. the SFM was trained for each emotion pair, each time slice and each left-out-participant individually (40 vs. 40 samples) and tested on the left-out-participant (8 vs. 8 samples).

First Iteration of a Support Feature Machine A single run SFM shows a clear time-dependent behaviour (see Figures 6.15 and 6.16). For almost all emotion pairs the number of relevant features is largest within the pre-emotional (-2 s) and the last post-emotional (30 s) phase. Starting with the first intra-emotional phase the number of features rapidly decreased and reached its minimum between 6 s (e.g. for *joy* vs. *anger*) and 18 s (e.g. for *joy* vs. *fear*). The number of features remained low for the rest of the intra-emotional phase — except for some outliers, e.g. at 14 s in *anger* vs. *disgust* — and increased again in the first post-emotional phase. The same behaviour is observed if we average across each emotion or across all pairs. The leave-one-participant-out cross-validation error of the SFM shows a very similar behaviour (see Figure 6.16). Although we only consider the error rate of a single-run SFM on very limited data, we observe a smoothly decreasing error rate with a minimum between 6 s and 18 s (except for two outliers, *disgust* vs. *fear* and *fear* vs. *sadness*). The error rate remains low throughout the remaining intra-emotional phase and increased in the post-emotional phase to reach chance level at 30 s. Again, the averaged results show the same tendencies.

Obviously, the SFM is suited to extract the causalities of the experimental design. There is no detectable information in the pre-emotional phase; affective information increases from the beginning of the intra-emotional phase is most prominent after 6 to 10 s, and subsides in the post-emotional phase. The slightly better than chance performance in the pre-emotional phase may be due to the experimental setup as the participants had to experience each emotion

	Emotional phase/Time slice								
	Pre			Intra					Post
Emotion pair	-2 s	2 s	6 s	10 s	14 s	18 s	22 s	26 s	30 s
Joy vs. Anger	10.3	7.3	**3.2**	4.2	3.7	3.7	5.5	10.8	11.3
Joy vs. Disgust	11.7	8.7	**6.0**	6.8	7.3	6.7	7.8	10.0	9.2
Joy vs. Fear	11.3	10.0	6.5	4.8	4.0	**3.7**	6.3	8.2	13.3
Joy vs. Sadness	10.2	7.0	6.2	4.7	**4.2**	4.3	5.7	10.3	11.7
Anger vs. Disgust	12.2	6.7	**5.3**	5.7	8.2	6.5	8.8	12.3	11.2
Anger vs. Fear	12.2	10.2	6.7	7.5	5.8	**5.2**	9.2	11.3	11.2
Anger vs. Sadness	10.5	9.3	6.7	**6.0**	7.3	7.7	10.2	10.8	11.5
Disgust vs. Fear	9.5	7.0	**5.2**	5.7	5.5	6.0	7.0	8.7	11.0
Disgust vs. Sadness	12.3	7.2	**6.3**	6.8	7.7	6.8	8.7	10.5	11.5
Fear vs. Sadness	10.3	8.5	9.0	8.3	**7.7**	8.5	11.0	12.2	11.3
Joy vs. any	10.9	8.2	5.5	5.1	4.8	**4.6**	6.3	9.8	11.4
Anger vs. any	11.3	8.4	**5.5**	5.8	6.2	5.8	8.4	11.3	11.3
Disgust vs. any	11.4	7.4	**5.7**	6.2	7.2	6.5	8.1	10.4	10.7
Fear vs. any	10.8	8.9	6.8	6.6	**5.8**	5.8	8.4	10.1	11.7
Sadness vs. any	10.8	8.0	7.0	**6.5**	6.7	6.8	8.9	11.0	11.5
any vs. any	11.1	8.2	6.1	6.0	6.1	**5.9**	8.0	10.5	11.3

Figure 6.15: Time slice-dependent number of features for pairwise emotion analysis and leave-one-participant-out cross-validation. Shown are the average numbers of obtained features for all emotion pairs and the averaged results for each emotion and across all pairs. The minimum number of features within each row is highlighted in boldface.

4 times in a row. Thus, they knew in advance which emotion to express. So in this phase, the SFM might extract information on how the brain prepares the expression of affective states.

Repetitive Support Feature Machine Finally, we used the repetitive SFM to further analyse the time-dependent distribution of discriminative information. We used the spatially once subsampled input data (*joy* vs. *sadness*, 5174 features) and trained the repetitive SFM for each time slice in the leave-one-participant-out fashion. The average number of relevant features (see Figure 6.17) confirms the behaviour of a single-run SFM: Throughout the whole sequence of SFMs the number of features is large in the pre-emotional phase, decreases to reach a minimum at 14 to 18 s and increases again in the post-emotional phase (22 s). However, the slope is very similar for all time slices — first, a steep increase within the first few repetitions, then, a slower

	Emotional phase/Time slice								
	Pre			Intra					Post
Emotion pair	-2 s	2 s	6 s	10 s	14 s	18 s	22 s	26 s	30 s
Joy vs. Anger	0.30	0.36	**0.14**	0.20	0.17	0.18	0.23	0.51	0.51
Joy vs. Disgust	0.50	0.31	0.41	0.36	0.31	**0.30**	0.48	0.51	0.50
Joy vs. Fear	0.32	0.34	0.23	**0.17**	0.18	0.24	0.28	0.33	0.43
Joy vs. Sadness	0.41	0.32	0.28	0.25	**0.18**	0.23	0.24	0.44	0.50
Anger vs. Disgust	0.46	0.36	**0.16**	0.21	0.32	0.29	0.29	0.54	0.41
Anger vs. Fear	0.49	0.39	0.35	0.31	0.31	**0.28**	0.55	0.42	0.57
Anger vs. Sadness	0.36	0.38	0.28	**0.22**	0.32	0.36	0.50	0.41	0.66
Disgust vs. Fear	0.36	0.33	0.35	0.32	0.33	0.30	**0.25**	0.44	0.32
Disgust vs. Sadness	0.45	0.36	0.33	0.29	0.30	**0.20**	0.40	0.45	0.52
Fear vs. Sadness	0.41	**0.27**	0.29	0.47	0.44	0.48	0.45	0.39	0.49
Joy vs. any	0.38	0.34	0.26	0.24	**0.21**	0.24	0.31	0.45	0.48
Anger vs. any	0.40	0.37	**0.23**	0.23	0.28	0.28	0.39	0.47	0.54
Disgust vs. any	0.44	0.34	0.31	0.30	0.32	**0.27**	0.35	0.48	0.44
Fear vs. any	0.40	0.33	**0.31**	0.32	0.32	0.33	0.38	0.39	0.45
Sadness vs. any	0.41	0.33	**0.30**	0.31	0.31	0.32	0.40	0.42	0.54
any vs. any	0.41	0.34	0.28	**0.28**	0.29	0.29	0.37	0.44	0.49

Figure 6.16: Time slice-dependent leave-one-participant-out cross-validation error for pairwise emotion analysis. Shown are the mean error rate for all emotion pairs, the average error rate for each emotion, and the average rate across all pairs. The minimum error rate within each row is highlighted in boldface.

increase until approximately $^2/_3$ of all features have been discarded, and finally again a steeper increase.

The time slice-dependent test errors show how affective information evolves over time (see Figure 6.18). In the pre-emotional slice (-2 s) the initial test error is slightly less than chance, i.e. the data contains almost no discriminative information. The very same behaviour is observed in the first intra-emotional slice (2 s). A significant amount of discriminative information seems to be present from the second intra-emotional slice (6 s) onwards up to the first post-emotional slice (22 s). The errors get smaller in the beginning and reach a stable level. The smallest initial error rate is obtained at 18 s. Thus, affective brain states are best separable approximately 18 s after onset of the emotional period. Considering a latency of the BOLD response of 6 s, this means that affective brain states are actually best separable after 12 s. Although the initial error rate does not significantly change from 14 s to 18 s, we observe that more features carry

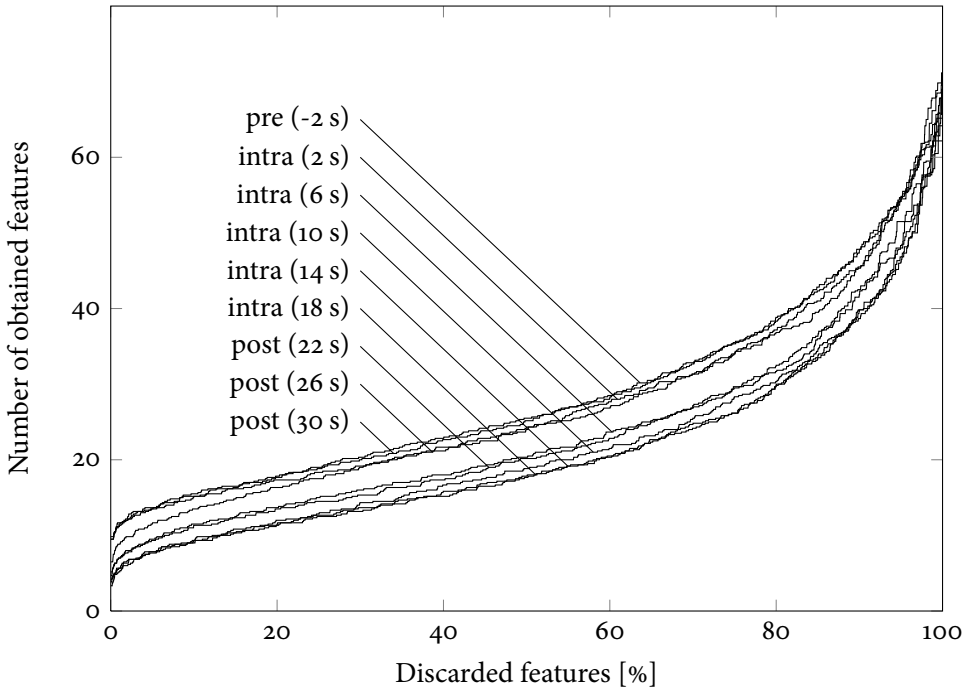


Figure 6.17: Average number of obtained features for each time slice (*joy* vs. *sadness*).

discriminative information, i.e. the error rate converges much slower to chance level. Thus, besides better separability we observe a higher degree of redundancy. Affective information spreads across larger regions but the accuracy does not further increase. At 18 s almost all voxels of the brain carry emotion-specific information. In the next time slice, the amount of information decreases, and in the second and third post-emotional slices almost no information seems to be present.

However, this procedure also reveals some deficiencies of our approach. In the second and third post-emotional slices (26 s and 30 s) the test error is almost constant but slightly better than chance (see Figure 6.18, bottom middle and bottom right). This systematic deviation from chance may have three reasons. First, the participants knew in advance which emotion to express and the same emotion was to be expressed multiple times. Thus, the input data might be biased in the pre- and post-emotional phases. This does not yet explain why affective information is omnipresent at a constant level. Second, the SFM might in certain scenarios be unable to identify the most informative features. Finally, accuracies better than chance might be due to the bias of cross-validation. Such too optimistic cross-validation rates may either be avoided by *nested* cross-validation [Varma and Simon, 2006] or the bias may be

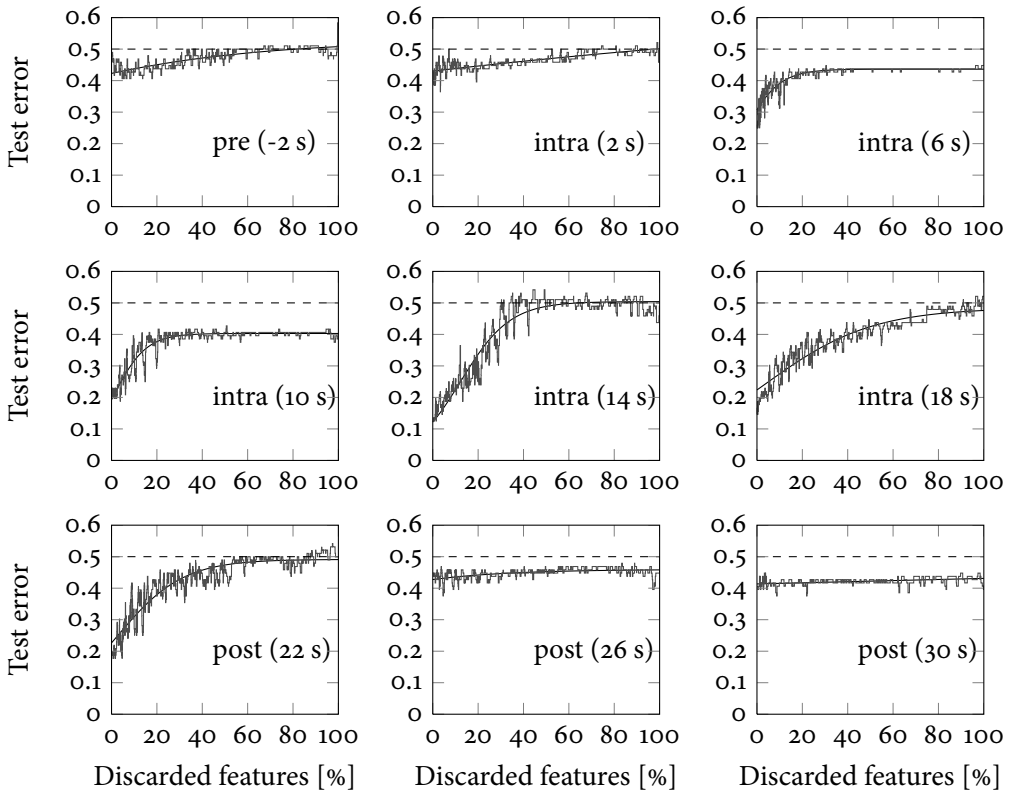


Figure 6.18: Time slice-dependent test error of an SVM on the irrelevant features for *joy* vs. *sadness*. Shown are the individual error curves and the fitted sigmoid curve.

corrected [Tibshirani and Tibshirani, 2009]. However, none of the proposed correction mechanisms can be used for our approach. We do not use cross-validation to optimise a certain parameter, but to obtain an estimate of the information content. Additionally, individual SFMs may not be compared across participants as they not necessarily use the same input data — for some participants certain features may have been discarded earlier than for others. In total, the obtained error curves are biased but convergence to chance level can be expected.

Finally, we visually compare the most discriminative time slice-dependent regions to those obtained on the whole trial beta images (see Figure 6.19). Time slice-dependent regions (red) and whole trial results (green) show the largest overlap (yellow) between 10 and 22 s. In the pre-emotional slice (-2 s) and in the second and third post-emotional slices (26 s and 30 s) there are almost no overlaps visible (only red or green regions, no yellow regions). With the onset of

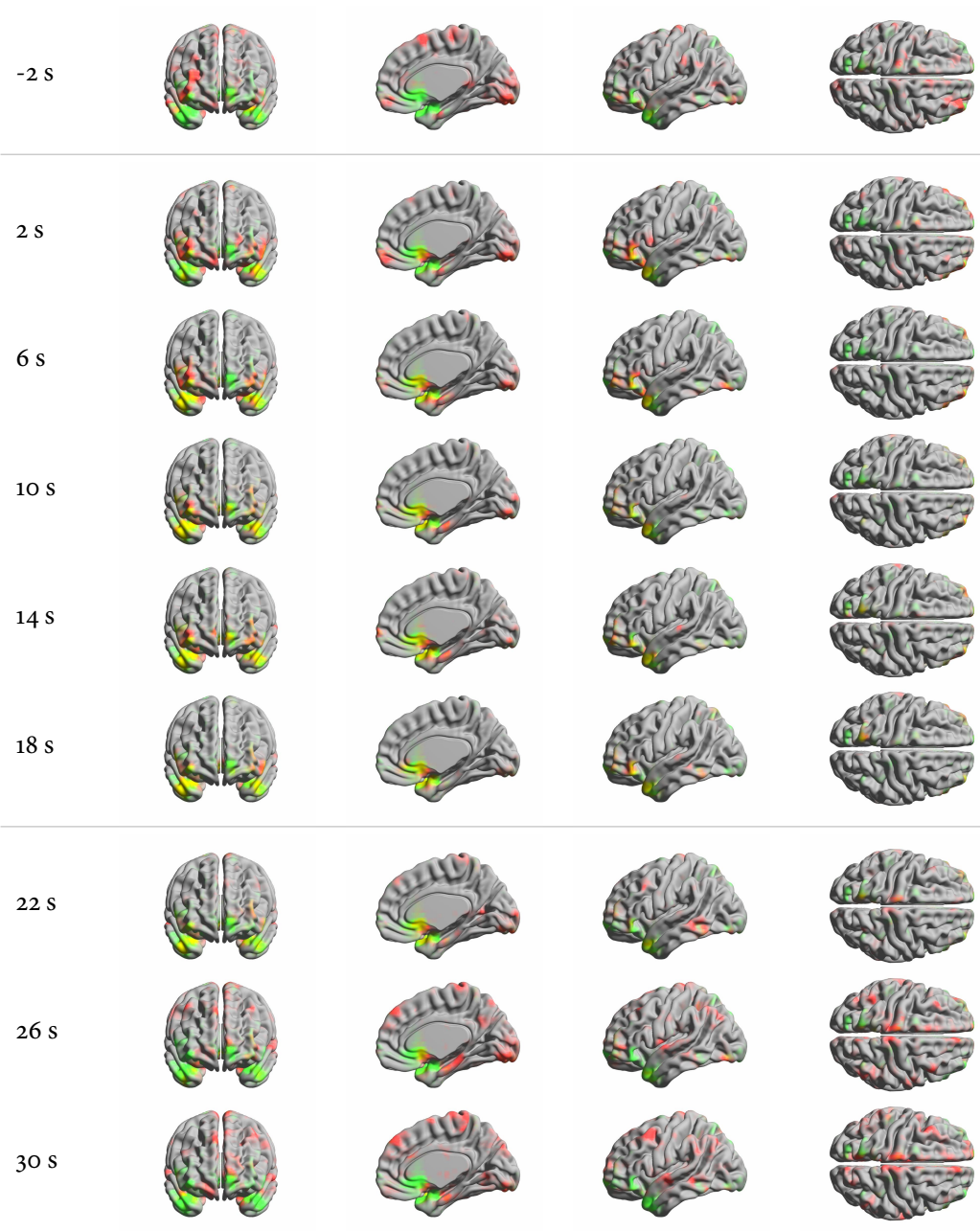


Figure 6.19: Time slice-dependent distribution of the most relevant voxels. Shown are the 2.5% most relevant voxels for all time slices (rows) and different brain views (columns) for *joy* vs. *sadness* in red. Voxels obtained in the per-trial analysis (see Chapter 6.4.1) are coloured in green, overlapping regions are yellow.

the trial the regions start to converge, more and more regions overlap (become yellow). With the end of the trial, the overlap again decreases.

In sum, we have shown how affective information in the human brain evolves over time. The separability — the largest achievable accuracy — as well as the redundancy increase over time. We observe that redundancy may further increase even if the separability remains constant. It seems that affective information is first present only in small brain regions but spreads out to finally involve almost the whole brain. However, the strength of this effect depends on which emotions we compared. Some emotions are less well separable, so separability and redundancy might be less well visible than for *joy* vs. *sadness*.

6.4.4 Downsampling Analysis

So far, we analysed brain activity data either on the complete voxel set or on the once downsampled data to improve runtime. The relevant regions obtained from the *buttonpress* dataset form two large clusters and only few minor regions. Thus, we expect the information content to be similarly present if the data is further downsampled. However, the minimum required resolution to provide a certain accuracy is unknown. On the *emotion* dataset, the relevant regions are distributed almost across the whole brain, but still they form clusters. Here, the affective information content may degrade if the resolution is further reduced. The following analysis aims to find the minimum required resolution to still obtain meaningful results based on the SFM. We did the same analysis for the *buttonpress* and the *emotion* dataset. The input data was downsampled three times (d = 50989, 6362, 791, 98 for *buttonpress* and d = 41642, 5222, 661, 80 for *emotion*). For the first level, the input data was simply subsampled as the input data had already been low-pass filtered in the preprocessing. The second and third downsampled images were obtained using the Gaussian pyramid based method with linear boundary extrapolation as described in Chapter 5. We stopped at level 3, as further downsampling would make linear separation infeasible.

Single-run Support Feature Machine For the *buttonpress* dataset, the number of obtained features and the leave-one-participant-out cross-validation error was almost constant on the first three pyramid levels but significantly increased at level 3 (see Figure 6.20). The behaviour for the *emotion* dataset was more diverse. For *joy* vs. *anger* we observe the same increase on the third level. Other emotion pairs showed a smooth performance decrease, e.g. *joy* vs. *disgust* or *fear* vs. *sadness*. The number of features may even remain almost constant, e.g. for *disgust* vs. *sadness*.

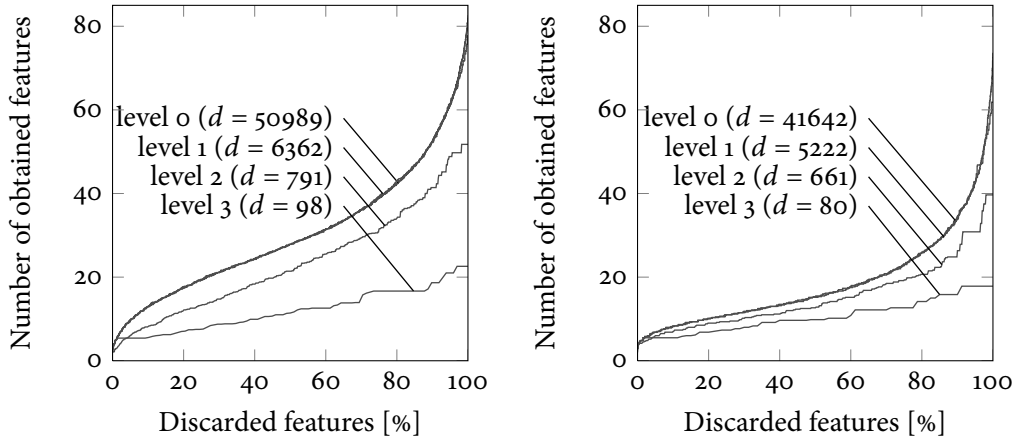
	Pyramid level (relevant features / test error)			
Dataset	0	1	2	3
buttonpress	*(d=50989)*	*(d=6362)*	*(d=791)*	*(d=98)*
left vs. right	2.0 / 0.03	2.4 / 0.09	2.2 / 0.03	5.4 / 0.17
emotion	*(d = 41642)*	*(d = 5222)*	*(d = 661)*	*(d = 80)*
Joy vs. Anger	2.3 / 0.16	3.2 / 0.19	2.0 / 0.10	4.7 / 0.20
Joy vs. Disgust	4.8 / 0.33	6.2 / 0.29	6.5 / 0.29	8.2 / 0.38
Joy vs. Fear	3.5 / 0.19	4.3 / 0.27	3.0 / 0.14	4.5 / 0.14
Joy vs. Sadness	4.0 / 0.23	3.5 / 0.26	4.7 / 0.27	5.5 / 0.34
Anger vs. Disgust	5.3 / 0.25	5.7 / 0.23	4.3 / 0.12	6.8 / 0.23
Anger vs. Fear	7.5 / 0.24	6.7 / 0.24	6.8 / 0.23	8.2 / 0.24
Anger vs. Sadness	6.8 / 0.34	6.3 / 0.43	7.3 / 0.38	10.3 / 0.30
Disgust vs. Fear	4.7 / 0.28	5.2 / 0.31	5.3 / 0.17	6.5 / 0.22
Disgust vs. Sadness	8.0 / 0.32	7.0 / 0.31	7.2 / 0.23	7.2 / 0.25
Fear vs. Sadness	7.5 / 0.42	8.3 / 0.44	8.7 / 0.40	11.5 / 0.41

Figure 6.20: Performance of a single-run SFM from high to low resolution input data. Shown are the number of obtained features and the leave-one-participant-out cross-validation error for each scenario.

Repetitive Support Feature Machine We evaluated the repetitive approach on the *buttonpress* task and for the *joy* vs. *sadness* classification task by successively training SFMs and discarding features for each of the pyramid levels. The results show that the number of features extracted on the original data (level 0) almost exactly matches the number of features obtained on the once downsampled data (see Figure 6.21). Thus, we conclude that the information content is the same in level 0 and level 1. So, our approach to always use the once downsampled data in all previous experiments was valid. Further downsampling changed the slope of the curves; the number of obtained features becomes less on average.

For both tasks and four pyramid levels, we visualised the distribution of relevant voxels (see Figures 6.22 and 6.23) by arranging them first in the downsampled brain and upsampling this low-resolution brain again to full resolution. This causes the borders between relevant and irrelevant regions to slightly blur.

For the *buttonpress* dataset, the relevant regions almost exactly match for all resolution levels except for some minor clusters on level 0 that are no longer represented by relevant voxels on level 3 (see Figure 6.22). We had to include the 5% most relevant voxels to obtain reasonable results — and could not only use 2.5% as in all previous experiments — because this is the lowest



(a) *buttonpress* (b) *joy* vs. *sadness*

Figure 6.21: Number of obtained features for different resolution levels for the *buttonpress* task (a) and to distinguish *joy* vs. *sadness* (b).

percentage that is larger than the percentage of features obtained by a single-run SFM. Although only 98 voxels were used on level 3, still the precentral and postcentral gyri are found to be relevant. Note, the high-resolution brain volume shows the cerebrum but not the cerebellum. However, relevant cerebellar regions are adjacent to the cerebrum, and, therefore, red regions are visible on the surface of the cerebrum. We conclude that localised brain activity, such as in motor tasks, may be decoded even on the thrice downsampled volume data. For such tasks, it might be of interest to set up the fMRI scanner to acquire images with a low spatial resolution but with a higher temporal resolution. Thus, we might be able to further analyse the temporal evolution of brain activity data.

In the discrimination of *joy* vs. *sadness* we observe a large overlap of relevant voxels on the original data and the once and twice downsampled volumes (see Figure 6.23). However, the distribution changes significantly on level 3 — more relevant voxels in the left frontal lobe but fewer relevant voxels in the orbital part of the inferior frontal gyrus. Thus, we conclude that affective information may be decoded on the twice downsampled data (661 voxels) almost equally well as on the original data.

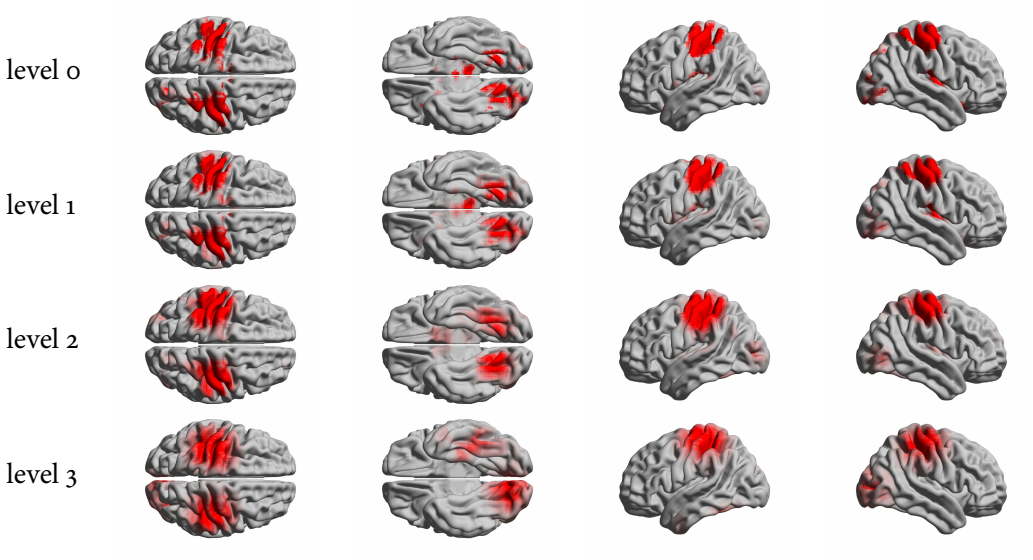


Figure 6.22: Most relevant regions on downsampled brain volume data in the *buttonpress* task. Shown are the 5% most relevant voxels for 4 pyramid levels.

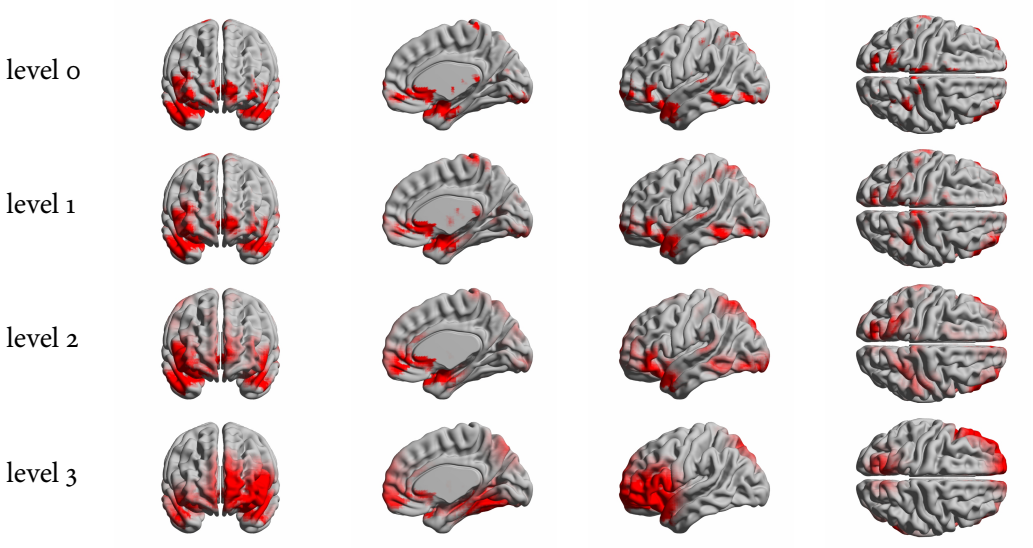


Figure 6.23: Most relevant regions in downsampled brain volume data to discriminate *joy* vs. *sadness*. Shown are the 7.5% most relevant voxels for 4 pyramid levels.

6.5 Conclusions

To further demonstrate the practical relevance of the SFM, we evaluated its capability to decode human brain states from fMRI data. For localised brain activity, the repetitive SFM is able to identify similar regions as mass univariate statistics and additionally provides an estimate of the number of informative voxels. We assume that the remaining voxels do no longer carry information if a linear SVM trained on these voxels performs on chance level. According to this measure, about 35% of all brain voxels carry information on whether a button was pressed with the left or the right thumb. The most relevant regions are located in the *precentral* and the *postcentral gyri* and in the *cerebellum*, which is in line with classical neuroscientific research. Thus, we conclude that the SFM is well suited to decode localised human brain states. But, as it is an intrinsically multidimensional method it may also be suited to decode complex brain activity such as affective states where mass univariate methods will fail.

Affective states were adressed in a series of experiments starting with a randomised support vector approach to select relevant features. We found that an SVM trained on a few randomly selected features — sometimes a single randomly selected feature — performs on average better than chance in discriminating two emotions when evaluated with leave-one-participant-out cross-validation. Further, with the repetitive SFM we identified the smallest set of voxels that allows decoding. In pairwise emotion analysis, the test errors differed significantly, e.g. from 16% with 2.3 features for *joy* vs. *anger* to 42% with 7.5 features for *fear* vs. *sadness*. In some cases, up to 80% of the brain carried discriminative information. Thus, we conclude that affective states are represented in whole brain activity patterns. However, some regions are more prominent than others — especially the orbital part of the inferior frontal gyrus was consistently found to be relevant for decoding across all emotion pairs. In a one-vs.-all emotions analysis, we observed the same tendencies, however, more features were found to be relevant than in the pairwise emotion analysis and the prediction accuracies were slightly worse.

The *emotion* dataset even allows to address the time-dependent behaviour of affective states. For a 20 s emotional phase, the repetitive SFM indicates that emotions may be best decoded between 6 and 18 s after the onset of the emotion. During the emotional phase not only separability increases but also information redundancy. Thus, after about 18 s almost the whole brain carries information to discriminate two emotions. In the post-emotional phase the affective information subsides and quickly reaches chance level again.

Finally, we assessed the location of the discriminative regions on different spatial resolutions. The previously proposed Gaussian pyramid based technique was used to obtain downsampled brain volumes. For localised brain activity, even the three times downsampled data contained sufficient information to discriminate motor tasks — the relevant regions as obtained by the

repetitive SFM were consistently found throughout all pyramid levels. For affective states, the data may be downsampled only twice to preserve the discriminative information. For both tasks, it seems promising to analyse the volume data on a lower spatial resolution to allow for fMRI data acquisition with a higher temporal resolution.

All these findings qualify the support feature machine with all its extensions as a comprehensive method for analysing brain activity data.

To know that one does not know, is the gift of a superior spirit. Not to know and to think that one does know, is a mistake. To know that this is a mistake, keeps one from making it.

From the movie
«Le mépris»
directed by Jean-Luc Godard

7 Conclusions

This thesis addressed three major issues. We have provided novel insight into the behaviour of high-dimensional small sample size data, we introduced the support feature machine as a novel method for classification with the least number of features, and we have applied this method in the field of cognitive neuroscience to analyse human brain activity data. Finally, we want to summarise the major findings, review the benefits of the proposed methods and address open questions, issues and promising directions of further work.

Machine learning methods provide valid results as long as large sample sizes in connection with comparatively low dimensionality are given. However, in practical applications such as the analysis of biological or medical data, we often face an inverse situation: Extremely few data points, for which it is impossible to significantly increase the sample size, and a high-dimensional feature space resulting from massively parallel data acquisition. Such data is prone to artefacts such as distance concentration, hubness, and incidental separability. Machine learning and validation methods may become unreliable. In the limit, for infinite dimensional data, very often all samples are located on the vertices of a regular simplex. Our evaluations related to leave-one-out cross-validation for support vector machines provide characteristics to decide whether a finite dimensional dataset is prone to such infinite-like unintuitive behaviour. Although we did only focus on the support vector machine, any metric-based classifier may display the same weakness. However, there is some evidence that using the one-norm — or even the zero-norm — instead of the standard Euclidean norm for measuring distances might reduce high-dimensional artefacts.

The main insight from all findings on high-dimensional small sample size experiments is the necessity to reduce dimensionality significantly wherever possible — best practise would be to not include irrelevant features in the training data at all. As this is feasible only in rare cases, feature selection mechanisms are necessary in a preprocessing phase or they may be included into the entire learning procedure. However, we have shown that the expressive

power — the VC-dimension — of combined feature selection and classification is still larger than the expressive power using only the intrinsically relevant features. Thus, although we apply feature selection we cannot expect to achieve the same prediction accuracy as if no irrelevant features were included in the input data at all.

Inspired by the SVM-based feature selection method of WESTON et al., we proposed the support feature machine with various extensions to deal with outliers, unbalanced datasets and redundant features. Both theoretically and empirically, we showed that the SFM is better suited — compared to WESTON's method — to identify the minimal set of relevant features. Besides dimensionality reduction, the paradigm of iteratively minimising the weight vector's one-norm in order to minimise its zero-norm should reduce the influence of high-dimensional artefacts as mentioned above.

The SFM may be implemented using commercial or publicly available linear programming solvers, but runtime requirements differ by orders of magnitude between solvers and alternative linear program formulations. We proposed an a-priori choice of formulation and solver depending on sample size and dimensionality. However, a method explicitly tuned to the SFM problem could outperform even the best general purpose solver. It will be beneficial to design a dedicated algorithm to reduce the overhead of general purpose solvers for data reorganisation, preprocessing or convergence checks. An in-depth analysis of the solver-specific differences might reveal why some are better suited than others and which linear program formulation is best suited — maybe others than the two proposed exist.

The experiments on artificial data and the real-world *leukemia* dataset showed that the SFM can identify relevant features very effectively and may improve the generalisation performance significantly with respect to an SVM without feature selection. Even an exponentially increasing number of irrelevant features does not cause a significant performance drop.

At this point, we shall also discuss why we exclusively focused on linear classifiers and whether the support feature machine may be extended to arbitrarily shaped decision borders. First, we focused on linear classifiers just because the data we address — biological, medical or neuroscientific data with many dimensions but few samples — does barely provide sufficient samples to determine all degrees of freedom of a linear decision surface. So, classifiers with even more expressive power that allow non-linear decision surfaces would be even less reliable. Thus, according to the principle of structural risk minimisation, we minimise the guaranteed risk by limiting the expressive power. Second, non-linear classifiers generally make use of kernel functions that implicitly use linear classifiers in a higher dimensional kernel space. It is often hard to interprete the results obtained in the kernel space with respect to the original input space. However, this is exactly what we aim for in real-world problems — to obtain meaningful results that are interpretable with respect to the original measurements. Here, non-linearity

would render interpretability unfeasible. Finally, the obtained results on real-world data suggest that linear classifiers have sufficient expressive power to obtain accuracies close to 100%. One reason for this might be the large observed redundancy of real-world data, i.e. information often seems to be encoded in several feature subsets. Thus, compared to non-redundant datasets, the probability is larger to find features that allow linear separation.

Besides the support feature machine as the main contribution of this thesis, we also proposed a supplementary method to remove illumination inhomogeneities from texture images based on the Gaussian pyramid. With standard filtering using the replicate or circular boundary condition, the resulting images would show artefacts at the image borders, mostly visible when stitching images together. To avoid these artefacts we proposed a framework that allows arbitrary boundary extrapolation with linear and polynomial extrapolation being the most promising to remove typical illumination gradients, such as natural vignetting, from real-world images. At first sight, both methods — the support feature machine and the Gaussian pyramid — seem to be unrelated. However, the latter may be used as a preprocessing step when the SFM is applied to images, i.e. where the input features are organised on a regular 2-dimensional or 3-dimensional grid as it is the case for fMRI data.

On such data, we used the SFM to decode human brain states. We found that the SFM identifies similar regions as mass univariate methods do, and it allows to quantify task-specific information. For a simple motor task, we observed up to 35% of all voxels of the brain to carry task-specific information. As the SFM is an intrinsically multivariate method, it is also qualified to analyse more complex fMRI data, such as affective brain states, where univariate methods are not appropriate. The decoding performance highly varies between emotions — *joy* seems to be best separable from all other emotions. Additionally, we observed that affective information is encoded in whole brain patterns with a large degree of redundancy. For some emotion pairs, almost the whole brain contains discriminative information. In a time-dependent analysis, we found the separability of emotions to improve over-time, but, besides, redundancy of the encoded information increased. Thus, at least two attributes characterise affective information — quantity and redundancy. To assess redundancy, we decoded motor tasks and emotional brain states on downsampled data as obtained by the aforementioned Gaussian pyramid-based method. The decoding accuracies remain almost the same for once and twice downsampled data; for motor tasks the data may even be downsampled three times such that the brain is sufficiently well represented by only 98 voxels.

These findings are even more impressing if we take into account the possible error sources that would degrade prediction accuracy. First, the input data contained only very few samples from a low number of participants (motor task: 12 participants, 2 classes, 4 samples per class; affective information: 6 participants, 5 classes, 8 samples per class). Thus, single outliers strongly

affect the prediction performance. Second, preprocessing aims to spatially align the brain data from different participants. Yet, such corrections can never perfectly normalise the brain data, i.e. the same voxel may still vary slightly in location across brains. Third, the support feature machine as a tool for decoding brain states and for measuring the amount and location of discriminative information is not perfect and may be affected by outliers and artefacts. However, even with all the above deficiencies we were able to decode emotional brain states with high accuracy and we found large regions of the brain to carry information. Thus, we expect the true information content to be even greater, wider distributed and more redundant than what we obtained from our experiments.

In total, our findings qualified the support feature machine as a universal method for feature selection especially suited in high-dimensional small sample size scenarios. Results obtained with the SFM on human brain data support the hypothesis that affective information is encoded in whole brain patterns with a large degree of redundancy.

Still, many open questions remain. How can the SFM be further improved with respect to prediction accuracy, feature selection correctness and runtime performance? May the SFM be implemented by a simple algorithm without the need of complicated linear programming solvers? How does the SFM perform with respect to other feature selection methods and benchmarks? Can we prove or disprove that the SFM is optimal according to any performance measure? How does a soft SFM perform on brain activity data — we omitted elaborate evaluations here due to runtime considerations? What other types of brain activity data may the SFM be used for? Does the SFM indeed provide novel insight into how information is processed in the human brain or does it only confirm already known facts? Which are the alternatives to independently verify the findings obtained by the SFM on FMRI data? Is it possible to use the SFM for more complex time-dependent mindreading tasks? Is mindreading a desirable technology at all and shouldn't we be more concerned about ethic issues of such a mentally invasive method? What comes after the odyssey?

Descended from the gods! Ulysses, cease;
Offend not Jove: obey, and give the peace.

«Odyssey», Homer,
translated by Alexander Pope

Bibliography

[Adelson et al., 1984] Adelson EH, Anderson CH, Bergen JR, Burt PJ, and Ogden JM. *Pyramid methods in image processing.* RCA Engineer, 29(6):33–41, 1984.

[Adini et al., 1997] Adini Y, Moses Y, and Ullman S. *Face recognition: the problem of compensating for changes in illumination direction.* IEEE TPAMI, 19:721–732, 1997.

[Aggarwal et al., 2001a] Aggarwal CC, Hinneburg A, and Keim DA. *On the surprising behavior of distance metrics in high dimensional space.* In *Proceedings of the 8th International Conference on Database Theory*, pages 420–434. Springer, 2001a.

[Aggarwal et al., 2001b] Aggarwal M, Hua H, and Ahuja N. *On cosine-fourth and vignetting effects in real lenses.* In *Proceedings of the 8th International Conference on Computer Vision*, pages 472–479. 2001b.

[Ahn et al., 2007] Ahn J, Marron JS, Muller KM, and Chi YY. *The high-dimension, low-sample-size geometric representation holds under mild conditions.* Biometrika, 94(3):760–766, 2007.

[Akbani et al., 2004] Akbani R, Kwek S, and Japkowicz N. *Applying support vector machines to imbalanced datasets.* In *In Proceedings of the 15th European Conference on Machine Learning*, pages 39–50. 2004.

[Anders et al., 2011] Anders S, Heinzle J, Weiskopf N, Ethofer T, and Haynes JD. *Flow of affective information between communicating brains.* NeuroImage, 54(1):439 – 446, 2011.

[Andersen and Andersen, 2000] Andersen ED and Andersen KD. *The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm*, pages 197–232. Kluwer Academic Publishers, 2000.

[Bartels and Zeki, 2004] Bartels A and Zeki S. *The chronoarchitecture of the human brain–natural viewing conditions reveal a time-based anatomy of the brain.* NeuroImage, 22(1):419–433, 2004.

[Bartlett and Shawe-Taylor, 1999] Bartlett P and Shawe-Taylor J. *Generalization performance of support vector machines and other pattern classifiers.* In *Advances in Kernel Methods: Support Vector Learning*, pages 43–54. MIT Press, Cambridge, MA, USA, 1999.

[Baucom et al., 2012] Baucom LB, Wedell DH, Wang J, Blitzer DN, and Shinkareva SV. *Decoding the neural representation of affective states.* NeuroImage, 59(1):718–727, 2012.

[Beer et al., 2006] Beer JS, Knight RT, and D'Esposito M. *Controlling the integration of emotion and cognition: The role of frontal cortex in distinguishing helpful from hurtful emotional information.* Psychological Science, 17(5):448–453, 2006.

[Bellman, 1961] Bellman RE. *Adaptive control processes — A guided tour.* Princeton University Press, Princeton, New Jersey, USA, 1961.

[Beyer et al., 1999] Beyer K, Goldstein J, Ramakrishnan R, and Shaft U. *When is "nearest neighbor" meaningful?* In *Proceedings of the 7th International Conference on Database Theory*, ICDT '99, pages 217–235. 1999.

[Bradley, 1997] Bradley AP. *The use of the area under the roc curve in the evaluation of machine learning algorithms.* Pattern Recognition, 30:1145–1159, 1997.

[Bradley and Mangasarian, 1998] Bradley PS and Mangasarian OL. *Feature selection via concave minimization and support vector machines.* In *Proceedings of the 15th International Conference on Machine Learning*, pages 82–90. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[Chaovalitwongse et al., 2007] Chaovalitwongse WA, Fan YJ, and Sachdeo RC. *Support feature machine for classification of abnormal brain activity.* In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 113–122. ACM, New York, NY, USA, 2007.

[Chapelle et al., 2002] Chapelle O, Vapnik V, Bousquet O, and Mukherjee S. *Choosing multiple parameters for support.* Machine Learning, 46(1):131–159, 2002.

[Charikar et al., 2000] Charikar M, Guruswami V, Kumar R, Rajagopalan S, and Sahai A. *Combinatorial feature selection problems.* In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 631–640. IEEE Computer Society, Washington, DC, USA, 2000.

[Chawla et al., 2002] Chawla NV, Bowyer KW, Hall LO, and Kegelmeyer WP. *SMOTE: Synthetic minority over-sampling technique*. Journal of Artificial Intelligence Research, 16:321–357, 2002.

[Chawla et al., 2004] Chawla NV, Japkowicz N, and Kotcz A. *Editorial: special issue on learning from imbalanced data sets*. ACM SIGKDD Explorations Newsletter, 6:1–6, 2004.

[Cover, 1965] Cover TM. *Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition*. IEEE Transactions on Electronic Computers, EC-14(3):326–334, 1965.

[Cristianini and Shawe-Taylor, 2000] Cristianini N and Shawe-Taylor J. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[Dantzig and Thapa, 2003] Dantzig G and Thapa M. *Linear Programming, Theory and Extensions*. Springer series in operations research. Springer, 2003.

[Daubechies et al., 2009] Daubechies I, Roussos E, Takerkart S, Benharrosh M, Golden C, D'Ardenne K, Richter W, Cohen JD, and Haxby J. *Independent component analysis for brain fMRI does not select for independence*. Proceedings of the National Academy of Sciences, 106(26):10415–10422, 2009.

[Davatzikos et al., 2005] Davatzikos C, Ruparel K, Fan Y, Shen DG, Acharyya M, Loughead JW, Gur RC, and Langleben DD. *Classifying spatial patterns of brain activity with machine learning methods: Application to lie detection*. Neuroimage, 28:663–668, 2005.

[Dehaene et al., 1998] Dehaene S, Clec'H GL, Cohen L, Poline JB, de Moortele PFV, and Le Bihan D. *Inferring behavior from functional images*. Nature Neuroscience, 1(7):549–550, 1998.

[Donoho and Elad, 2003] Donoho D and Elad M. *Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization*. Proceedings of the National Academy of Sciences of the United States of America, 100(5):2197–2202, 2003.

[Duda et al., 2001] Duda RO, Hart PE, and Stork DG. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2001.

[Efron, 1979] Efron B. *Bootstrap methods: Another look at the jackknife*. The Annals of Statistics, 7(1):1–26, 1979.

[Efron and Tibshirani, 1993] Efron B and Tibshirani RJ. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.

[Efros and Freeman, 2001] Efros AA and Freeman WT. *Image quilting for texture synthesis and transfer*. In *SIGGRAPH*, pages 341–346. ACM, New York, NY, USA, 2001.

[Elizondo, 2006] Elizondo DA. *The linear separability problem: some testing methods*. IEEE Transactions on Neural Networks, 17(2):330–344, 2006.

[François et al., 2007] François D, Wertz V, and Verleysen M. *The concentration of fractional distances*. IEEE Transactions on Knowledge and Data Engineering, 19:873–886, 2007.

[Friston et al., 1995] Friston K, Holmes A, Worsley K, Poline J, Frith C, and Frackowiak R. *Statistical parametric maps in functional imaging: A general linear approach*. Human Brain Mapping, 2:189–210, 1995.

[Furey et al., 2000] Furey TS, Cristianini N, Duffy N, Bednarski DW, Schummer M, and Haussler D. *Support vector machine classification and validation of cancer tissue samples using microarray expression data*. Bioinformatics, 16(10):906–914, 2000.

[Golub et al., 1999] Golub T, Slonim D, Tamayo P, Huard C, Gaasenbeek M, Mesirov J, Coller H, Loh M, Downing J, Caligiuri M, Bloomfield C, and Lander E. *Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring*. Science, 286(5439):531–537, 1999.

[Guyon et al., 2002] Guyon I, Weston J, Barnhill S, and Vapnik V. *Gene selection for cancer classification using support vector machines*. Machine Learning, 46:389–422, 2002.

[Hall et al., 2005] Hall P, Marron JS, and Neeman A. *Geometric representation of high dimension, low sample size data*. Journal of the Royal Statistical Society, 67(3):427–444, 2005.

[Hand and Till, 2001] Hand DJ and Till RJ. *A simple generalisation of the area under the roc curve for multiple class classification problems*. Machine Learning, 45:171–186, 2001.

[Haykin, 1998] Haykin S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[Haynes, 2011] Haynes JD. *Special issue on multivariate decoding and brain reading*. NeuroImage, 56, 2011.

[Haynes and Rees, 2005] Haynes JD and Rees G. *Predicting the orientation of invisible stimuli from activity in human primary visual cortex*. Nature Neuroscience, 8(5):686–691, 2005.

[Haynes and Rees, 2006] Haynes JD and Rees G. *Decoding mental states from brain activity in humans*. Nature Reviews Neuroscience, 7:523–534, 2006.

[Haynes et al., 2007] Haynes JDD, Sakai K, Rees G, Gilbert S, Frith C, and Passingham RE. *Reading hidden intentions in the human brain*. Current Biology, 17(4):323–328, 2007.

[He and Garcia, 2009] He H and Garcia EA. *Learning from imbalanced data*. IEEE Transactions on Knowledge and Data Engineering, 21:1263–1284, 2009.

[Holmes et al., 1997] Holmes A, Poline J, and Friston K. *Characterizing brain images with the general linear model*. In R Frackowiak, K Friston, C Frith, R Dolan, and J Mazziotta, editors, *Human Brain Function*, pages 59–84. Academic Press USA, 1997.

[Hosseini et al., 2011] Hosseini SMH, Mano Y, Rostami M, Takahashi M, Sugiura M, and Kawashima R. *Decoding what one likes or dislikes from single-trial fNIRS measurements*. NeuroReport, 22(6):269–273, 2011.

[Japkowicz, 2000] Japkowicz N. *Learning from imbalanced data sets: A comparison of various strategies*. In *Learning from Imbalanced Data Sets, Papers from the AAAI Workshop, Technical Report WS-00-05*, pages 10–15. AAAI Press, 2000.

[Jebara and Jaakkola, 2000] Jebara T and Jaakkola T. *Feature selection and dualities in maximum entropy discrimination*. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 291–300. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[Kearns and Ron, 1997] Kearns MJ and Ron D. *Algorithmic stability and sanity-check bounds for leave-one-out cross-validation*. Neural Computation, 11:152–162, 1997.

[Kim and Pollefeys, 2008] Kim SJ and Pollefeys M. *Robust radiometric calibration and vignetting correction*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(4):562–576, 2008.

[Kira and Rendell, 1992a] Kira K and Rendell LA. *The feature selection problem: Traditional methods and a new algorithm*. In *AAAI*, pages 129–134. AAAI Press and MIT Press, Cambridge, MA, USA, 1992a.

[Kira and Rendell, 1992b] Kira K and Rendell LA. *A practical approach to feature selection*. In *Proceedings of the 9th International Workshop on Machine Learning*, pages 249–256. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992b.

[Klement and Martinetz, 2010a] Klement S and Martinetz T. *A new approach to classification with the least number of features*. In *9th International Conference on Machine Learning and Applications*, pages 141–146. IEEE Computer Society, 2010a.

[Klement and Martinetz, 2010b] Klement S and Martinetz T. *The support feature machine for classifying with the least number of features*. In KI Diamantaras, W Duch, and LS Iliadis, editors, *ICANN (2)*, volume 6353 of *Lecture Notes in Computer Science*, pages 88–93. Springer, 2010b.

[Klement and Martinetz, 2011] Klement S and Martinetz T. *On the problem of finding the least number of features by L1-norm minimisation*. In T Honkela, editor, *Proceedings of the 21st International Conference on Artificial Neural Networks*, Lecture Notes in Computer Science 6791, pages 315–322. Springer, Heidelberg, 2011.

[Klement et al., 2008] Klement S, Madany Mamlouk A, and Martinetz T. *Reliability of cross-validation for SVMs in high-dimensional, low sample size scenarios*. In *Proceedings of the 18th International Conference on Artificial Neural Networks*, pages 41–50. Springer-Verlag, Berlin, Heidelberg, 2008.

[Klement et al., 2011] Klement S, Timm F, and Barth E. *Illumination correction for image stitching*. In *Proceedings of the International Conference on Imaging Theory and Applications*, volume 1, pages 81–86. INSTICC, 2011.

[Klement et al., 2013] Klement S, Anders S, and Martinetz T. *The support feature machine: Classification with the least number of features and its application to neuroimaging data*, 2013. Accepted.

[Kohavi, 1995] Kohavi R. *A study of cross-validation and bootstrap for accuracy estimation and model selection*. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1137–1145. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

[Kohavi and John, 1997] Kohavi R and John GH. *Wrappers for feature subset selection*. Artificial Intelligence, 97(1):273–323, 1997.

[Krauth and Mézard, 1987] Krauth W and Mézard M. *Learning algorithms with optimal stability in neural networks*. Journal of Physics A: Mathematical and General, 20(11):L745–L752, 1987.

[Kringelbach and Rolls, 2004] Kringelbach ML and Rolls ET. *The functional neuroanatomy of the human orbitofrontal cortex: evidence from neuroimaging and neuropsychology*. Progress in Neurobiology, 72(5):341–372, 2004.

[Lavine et al., 1988] Lavine BK, Jurs PC, and Henry DR. *Chance classifications by non-parametric linear discriminant functions*. Journal of Chemometrics, 2(1):1–10, 1988.

[LeCun et al., 1990] LeCun Y, Denker JS, and Solla SA. *Optimal brain damage*. In *Advances in Neural Information Processing Systems 2*, pages 598–605. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[Levin et al., 2004] Levin A, Zomet A, Peleg S, and Weiss Y. *Seamless image stitching in the gradient domain*. In *Proceedings of the 8th European Conference on Computer Vision*, pages 377–389. Springer-Verlag, 2004.

[Likar et al., 2001] Likar B, Viergever MA, and Pernuš F. *Retrospective correction of MR intensity inhomogeneity by information minimization*. IEEE Transactions on Medical Imaging, 20(12):1398–1410, 2001.

[Ling et al., 2003] Ling CX, Huang J, and Zhang H. *AUC: a statistically consistent and more discriminating measure than accuracy*. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 329–341. 2003.

[Lockhart and Winzeler, 2000] Lockhart DJ and Winzeler E. *Genomics, gene expression and dna arrays*. Nature, 405:827–36, 2000.

[MacKay, 2002] MacKay DJC. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 1st edition, 2002.

[Mangasarian, 1965] Mangasarian OL. *Linear and nonlinear separation of patterns by linear programming*. Operations Research, 13:444–452, 1965.

[Martinetz, 2004] Martinetz T. *MinOver revisited for incremental support-vector-classification*. In C Rasmussen, H Buelthoff, M Giese, and B Schoelkopf, editors, *DAGM 2004*, volume 3175 of *Lecture Notes in Computer Science*, pages 187–194. Springer Press, Heidelberg, 2004.

[Martinetz et al., 2005] Martinetz T, Labusch K, and Schneegass D. *SoftDoubleMinOver: A simple procedure for maximum margin classification*. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and their Applications, Part II*, pages 301–306. Springer-Verlag, Berlin, Heidelberg, 2005.

[Maszczyk and Duch, 2010a] Maszczyk T and Duch W. *Support feature machine for DNA microarray data*. In M Szczuka, M Kryszkiewicz, S Ramanna, R Jensen, and Q Hu, editors, *Proceedings of the 7th International Conference on Rough Sets and Current Trends in Computing*, volume 6086 of *Lecture Notes in Computer Science*, pages 178–186. Springer Press, Heidelberg, 2010a.

[Maszczyk and Duch, 2010b] Maszczyk T and Duch W. *Support feature machines: Support vectors are not enough*. In *International Joint Conference on Neural Networks*, pages 3852–3859. IEEE, 2010b.

[McKeown et al., 1998] McKeown MJ, Makeig S, Brown GG, Jung TP, Kindermann SS, Kindermann RS, Bell AJ, and Sejnowski TJ. *Analysis of fMRI data by blind separation into independent spatial components*. Human Brain Mapping, 6:160–188, 1998.

[McPherson et al., 2007] McPherson R, Pertsemlidis A, Kavaslar N, Stewart A, Roberts R, Cox DR, Hinds DA, Pennacchio LA, Tybjaerg-Hansen A, Folsom AR, Boerwinkle E, Hobbs HH, and Cohen JC. *A common allele on chromosome 9 associated with coronary heart disease*. Science, 316(5830):1488 –1491, 2007.

[Mehrotra, 1992] Mehrotra S. *On the implementation of a primal-dual interior point method*. SIAM Journal on Optimization, 2:575–601, 1992.

[Mourão Miranda et al., 2005] Mourão Miranda J, Bokde ALW, Born C, Hampel H, and Stetter M. *Classifying brain states and determining the discriminating activation patterns: Support vector machine on functional MRI data*. NeuroImage, 28(4):980–995, 2005.

[Mitchell et al., 2008] Mitchell TM, Shinkareva SV, Carlson A, Chang KMM, Malave VL, Mason RA, and Just MAA. *Predicting human brain activity associated with the meanings of nouns*. Science, 320(5880):1191–1195, 2008.

[Mukherjee et al., 1998] Mukherjee S, Tamayo P, Slonim D, Verri A, Golub T, Mesirov JP, and Poggio T. *Support vector machine classification of microarray data*. Technical report, AI Memo 1677, Massachusetts Institute of Technology, 1998.

[Ogden et al., 1985] Ogden JM, Adelson EH, Bergen JR, and Burt PJ. *Pyramid-based computer graphics*. RCA Engineer, pages 4–15, 1985.

[Pessoa and Padmala, 2007] Pessoa L and Padmala S. *Decoding near-threshold perception of fear from distributed single-trial brain activation*. Cerebral Cortex, 17:691–701, 2007.

[Platt, 1999] Platt JC. *Fast training of support vector machines using sequential minimal optimization*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[Provost, 2000] Provost F. *Machine learning from imbalanced data sets 101*. Proceedings of the AAAI-2000 Workshop on Imbalanced Data Sets, 2000.

[Radovanović et al., 2010] Radovanović M, Nanopoulos A, and Ivanović M. *Hubs in space: Popular nearest neighbors in high-dimensional data*. Journal of Machine Learning Research, 11:2487–2531, 2010.

[Raelson et al., 2007] Raelson JV, Little RD, Ruether A, Fournier H, Paquin B, Van Eerdewegh P, Bradley WEC, Croteau P, Nguyen-Huu Q, Segal J, Debrus S, Allard R, Rosenstiel P, Franke A, Jacobs G, Nikolaus S, Vidal JM, Szego P, Laplante N, Clark HF, Paulussen RJ, Hooper JW, Keith TP, Belouchi A, and Schreiber S. *Genome-wide association study for crohn's disease in the quebec founder population identifies multiple validated disease loci*. Proceedings of the National Academy of Sciences, 104(37):14747–14752, 2007.

[Raskutti and Kowalczyk, 2004] Raskutti B and Kowalczyk A. *Extreme re-balancing for SVMs: a case study*. SIGKDD Explorations Newsletter, 6:60–69, 2004.

[Rolland et al., 2000] Rolland JP, Vo V, Bloss B, and Abbey CK. *Fast algorithms for histogram matching: Application to texture synthesis*. Journal of Electronic Imaging, 9(1):39–45, 2000.

[Rosenblatt, 1958] Rosenblatt F. *The perceptron: a probabilistic model for information storage and organization in the brain*. Psychological Reviews, 65(6):386–408, 1958.

[Saeys et al., 2007] Saeys Y, Inza In, and Larrañaga P. *A review of feature selection techniques in bioinformatics*. Bioinformatics, 23(19):2507–2517, 2007.

[Samani et al., 2007] Samani NJ, Erdmann J, Hall AS, Hengstenberg C, Mangino M, Mayer B, Dixon RJ, Meitinger T, Braund P, Wichmann H, Barrett JH, König IR, Stevens SE, Szymczak S, Tregouet D, Iles MM, Pahlke F, Pollard H, Lieb W, Cambien F, Fischer M, Ouwehand W, Blankenberg S, Balmforth AJ, Baessler A, Ball SG, Strom TM, Brænne I, Gieger C, Deloukas P, Tobin MD, Ziegler A, Thompson JR, and Schunkert H. *Genomewide association analysis of coronary artery disease*. The New England Journal of Medicine, 357(5):443–453, 2007.

[Sauer, 1972] Sauer N. *On the density of families of sets*. Journal of Combinatorial Theory, Series A, 13(1):145 – 147, 1972.

[Schmahmann et al., 1998] Schmahmann JD, Doyon J, Mcdonald D, Holmes C, Lavoie K, Hurwitz AS, Kabani N, Toga A, Evans A, and Petrides M. *Three-dimensional MRI atlas of the human cerebellum in proportional stereotaxic space*. NeuroImage, 10:233–260, 1998.

[Slonim et al., 2000] Slonim DK, Tamayo P, Mesirov JP, Golub TR, and Lander ES. *Class prediction and discovery using gene expression data*. In *Proceedings of the 4th Annual International Conference on Computational Molecular Biology*, pages 263–272. ACM, New York, NY, USA, 2000.

[Stephens et al., 2010] Stephens GJ, Silbert LJ, and Hasson U. *Speaker–listener neural coupling underlies successful communication*. Proceedings of the National Academy of Sciences, 107(32):14425–14430, 2010.

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

[Tibshirani and Tibshirani, 2009] Tibshirani RJ and Tibshirani R. *A bias correction for the minimum error rate in cross-validation*. The Annals of Applied Statistics, 3(2):822–829, 2009.

[Tzourio-Mazoyer et al., 2002] Tzourio-Mazoyer N, Landeau B, Papathanassiou D, Crivello F, Etard O, Delcroix N, Mazoyer B, and Joliot M. *Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain*. NeuroImage, 15(1):273–289, 2002.

[Vanderbei, 2008] Vanderbei R. *Linear programming: foundations and extensions*. International series in operations research & management science. Springer, 2008.

[Vapnik, 1982] Vapnik V. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[Vapnik, 1999] Vapnik VN. *The Nature of Statistical Learning Theory*. Springer, 1999.

[Vapnik and Chervonenkis, 1971] Vapnik VN and Chervonenkis AY. *On the uniform convergence of relative frequencies of events to their probabilities*. Theory of Probability and its Applications, 16(2):264–280, 1971.

[Vapnik and Chervonenkis, 1982] Vapnik VN and Chervonenkis AY. *Necessary and sufficient conditions for the uniform convergence of means to their expectations*. Theory of Probability and its Applications, 26(3):532–553, 1982.

[Varma and Simon, 2006] Varma S and Simon R. *Bias in error estimation when using cross-validation for model selection*. BMC Bioinformatics, 7(1):91, 2006.

[Veropoulos et al., 1999] Veropoulos K, Campbell C, and Cristianini N. *Controlling the sensitivity of support vector machines*. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 55–60. 1999.

[Vul et al., 2009] Vul E, Harris C, Winkielman P, and Pashler H. *Puzzlingly high correlations in fmri studies of emotion, personality, and social cognition*. Perspectives on Psychological Science, 4(3):274–290, 2009.

[Wendel, 1962] Wendel J. *A problem in geometric probability*. Mathematics Scandinavia, 11:109–111, 1962.

[Weston et al., 2000] Weston J, Mukherjee S, Chapelle O, Pontil M, Poggio T, and Vapnik V. *Feature selection for SVMs*. In *Advances in Neural Information Processing Systems*. 2000.

[Weston et al., 2003] Weston J, Elisseeff A, Schölkopf B, and Tipping M. *Use of the zero-norm with linear models and kernel methods*. Journal of Machine Learning Research, 3:1439–1461, 2003.

[Yogananda et al., 2007] Yogananda PA, Murthy MN, and Gopal L. *A fast linear separability test by projection of positive points on subspaces*. In *Proceedings of the 24th international conference on Machine learning*, pages 713–720. ACM, New York, NY, USA, 2007.

[Zheng et al., 2009] Zheng Y, Lin S, Kambhamettu C, Yu J, and Kang S. *Single-image vignetting correction*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(12):2243–2256, 2009.

[Zhu et al., 2004] Zhu J, Rosset S, Hastie T, and Tibshirani R. *1-norm support vector machines*. In S Thrun, L Saul, and B Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.