TAMPERE UNIVERSITY OF TECHNOLOGY

FERDINAND KAISER

Robust Support Vector Machines For Implicit Outlier Removal

Master of Science Thesis

Examiners: Dr. Tech. Ari Visa
M.Sc. Mikko Parviainen
Examiners and topic approved in the Department of Signal Processing meeting on 7th November 2012

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
FERDINAND KAISER: Robust Support Vector Machines For Implicit Outlier Removal
Master of Science Thesis, 42 pages, 8 Appendix pages
November 2012
Major: Signal processing
Examiner: Dr. Tech. Ari Visa, M.Sc. Mikko Parviainen
Keywords: Support Vector Machine, Robustness, Truncated Hinge Loss Function

The support vector machine is a machine learning algorithm which has been successfully applied to solve classification problems since its introduction in the early 1990s. It is based on the work of Vladimir Vapnik on Statistical Learning Theory and is theoretically well founded. Following the discriminative approach, the SVM yields a classifier which separates two classes by a hyperplane. The training instances are classified according to the sign of their distance to the hyperplane. This hyperplane is defined by a small number of training instances such that the distance of the training instances of both classes to the hyperplane is maximized and the misclassification error is minimized. Hence the support vector machine belongs to the family of maximum margin classifiers. Since the support vector machine does not estimate the underlying class conditional distribution of the training instances, but instead uses them directly to construct the classifier, it is important that the training instances are sampled from the underlying class conditional distribution. If this is not the case because the training set is contaminated with outliers, the accuracy of the classifier defined by the support vector machine decreases. Based on this observation several approaches have been proposed to improve the robustness of the support vector machine against outliers in the training data. In this thesis we will discuss the class robust support vector machines which aim to make the standard support vector machine robust against noise by implicit outlier filtering. Those approaches are using the support vector machine to detect and remove outliers based on their position relative to the separating hyperplane. Since the success of those methods is only empirically proven, we conduct a thoroughly experimental study in order to determine under which conditions those robust methods can be applied in practice. We are especially interested if the additional parameter which controls the removal of outliers can be estimated from a training set which is contamined by outliers.

PREFACE

I would like to thank my thesis supervisor Dr. Tech. Ari Visa for all the good advices and insights during the thesis process. And of course I also want to thank my family and my girlfriend for their love and constant support.

Ferdinand Kaiser

Tampere, November 27, 2012

TABLE OF CONTENTS

1. Introduction 1
1.0.1 Thesis Outline 2
2. Theoretical Background 3
2.1 Supervised Learning 3
2.2 Statistical Learning Theory 7
2.2.1 Empirical Risk Minimization 7
2.2.2 Consistency 8
2.2.3 Capacity 9
2.2.4 Structural Risk Minimization 12
2.3 Optimization Theory 12
2.3.1 Lagrangian Duality 13
2.4 Support Vector Machines 15
2.4.1 Hard-margin SVM 15
2.4.2 Soft-margin SVM 20
2.4.3 Nonlinear SVM 22
2.4.4 Robustness of the SVM 23
3. Robust SVM approaches 25
3.1 The Influence of the Loss Function on the SVM 25
3.2 η-Hinge Loss Function 27
3.3 Truncated Hinge Loss Function 28
3.4 Integrated Outlier Filtering 30
4. Experiments 32
4.1 Data sets, Resampling and Preparation 33
4.2 Experimental Strategy 35
4.3 Implementation 36
5. Results 37
5.1 Discussion 41
6. Conclusion 42
A.Appendix 46

LIST OF FIGURES

2.1 Flowchart of supervised learning 4
2.2 A classification problem in two dimensional space 5
2.3 The convergence of expected and empirical risk 8
2.4 An overfitting example 9
2.5 Shattering in two dimensional space 10
2.6 The value of the VC bound depending on the VC dimension 11
2.7 Solving a convex and a non-convex in the primal and dual 14
2.8 The optimal separating hyperplane 17
2.9 Soft-margin SVM using a linear kernel 20
2.10 Non-linear soft-margin SVM using a RBF kernel 23

3.1 Four different loss functions 26
3.2 The filtering hinge loss function 30

4.1 Visualization of the data sets used in the experiments 34

5.1 Histograms of the accuracy measures 37
5.2 Results for the UCI Breast cancer dataset using a linear kernel 38
5.3 Results for the UCI Liver disorders and UCI Heart dataset using a linear kernel 39
5.4 Results for the UCI Fourclass dataset using a RBF kernel 40

LIST OF TABLES

4.1 Dimensionality and size of the training and test sets used for evaluation 33

A.1 Classification accuracies using the linear kernel on datasets contamined by class noise 47
A.2 Classification accuracies using the RBF kernel on datasets contamined by class noise 48
A.3 Classification accuracies using the linear kernel on datasets contamined by adversarial noise 49
A.4 Classification accuracies using the RBF kernel on datasets contamined by adversarial noise 50
A.5 Number of support vectors and F1-scores using the linear kernel on datasets contamined by class noise 51
A.6 Number of support vectors and F1-scores using the RBF kernel on datasets contamined by class noise. 52
A.7 Number of support vectors and F1-scores using the linear kernel on datasets contamined by adversarial noise. 53
A.8 Number of support vectors and F1-scores using the RBF kernel on datasets contamined by adversarial noise. 54

ABBREVIATIONS

CCCP	Constrained Concave-Convex Procedure
CV	Cross-Validation
ERM	Empirical Risk Minimization
KKT conditions	Karush-Kuhn-Tucker conditions
RBF	Radial Basis Function
SLT	Statistical Learning Theory
SRM	Structural Risk Minimization
VC-dimension	Vapnik-Chervonenkis dimension

1. INTRODUCTION

Learning from examples, also known as supervised learning, requires to infer a decision rule from examples. If we assume that some examples presented to the learner are erroneous or in other words outliers with respect to the other examples, the learner needs to be able to detect them. Suppose a human wants to learn how to discriminate between two different classes of flowers given a set of labeled examples. Based on some features of the flowers such as color or smell and the label provided by the teacher, a human finds a way how to distinguish between those two species effectively. If some of the examples are incorrectly labeled, a human would notice that those mislabeled examples are more similar to the other class than to the actual class label. Consequently those examples would be excluded from the learning process. This ability to detect training examples which are highly unlikely to occur and which contradict the general distribution of the data is essential in supervised learning, since real world data is often polluted by a certain amount of errors. Those errors are either introduced by mislabeling through the teacher as mentioned above, or by measurement errors during the acquisition of the features. The way a human would recognize outliers is based on certain assumptions. One assumption could be that the examples belonging to one class should not differ too much in their features. Consequently learning algorithms designed to be executed by computers need to include methods in order to identify erroneous training examples.

In this thesis we show how supervised learning is implemented by a specific learning algorithm, the support vector machine. Furthermore we present a specific class of approaches which have been proposed in order to make this learning algorithm robust against outliers in the training data. Different from the example above the robust methods we are focusing on do not rely on a specific assumption about the distribution of the data, but implicitly remove outliers during the training process of the classifier. In the experimental part of this thesis we evaluate how accurately the outliers are detected and under which conditions the robust methods improve the performance of the support vector machine. This evaluation is based on real-world datasets which are contamined with outliers. We measure the classification accuracy of the support vector machine and the robust approaches as well as the accuracy of outlier removal in order to compare the different classifiers.

1.0.1 Thesis Outline

This thesis is divided into six chapters. In chapter two we discuss the theoretical background of support vector machines, the different types of support vector machines as well as the robustness property of the standard support vector machine. Chapter three introduces the different approaches to integrate outlier removal into the training of the standard support vector machine by using alternative loss functions. In the fourth chapter the experimental setup is described, followed by the fifth chapter were the experimental results are discussed. The last chapter summarizes our findings.

2. THEORETICAL BACKGROUND

2.1 Supervised Learning

Learning machines which are trained based on a set of instances x and labels y belong to the family of supervised learning algorithms. Figure 2.1 explains the model of the supervised learning setting. The model consists of a generator G which samples instances x independently from a unknown fixed distribution $P(X)$ and a supervisor T which assigns an output value y to x according to a fixed unknown distribution $P(Y|X)$. The learning machine observes a set of training pairs $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_l, y_l)\}$. Based on the presented examples the learning machine selects a function $h : \mathcal{X} \to \mathcal{Y}$ from a class of functions $\mathcal{H}$ such as to approximate the output of T as good as possible for any given x. The function h is called the hypothesis. The process of choosing an appropriate hypothesis from a class of Hypothesis $\mathcal{H}$ is where the machine is learning [22] and is denoted as the training of the classifier.

Supervised learning algorithms differ in how learning is performed and how the training pairs are presented to the learner. It is distinguished between batch learning, online learning and active learning. In this thesis we will focus on batch learning which assumes the training set is fixed and all training samples are independently and identically distributed, that is the learning process has no influence on the composition of the training set.

Depending on the type of output values y the class of supervised learning problems is distinguished classification and regression problems. In classification problems the output space $\mathcal{Y}$ is a set of discrete values whereas regression problems deal with continuous outputs

$$h : \mathcal{X} \to \{-1, 1\} \ , \ \mathcal{X} = \mathbb{R}^n. \tag{2.1}$$

In classification problems the hypothesis is also denoted as decision function or just the classifier. The aim is to find a hypothesis which performs good not only on the training data but on all the data, that is the hypothesis should make as little

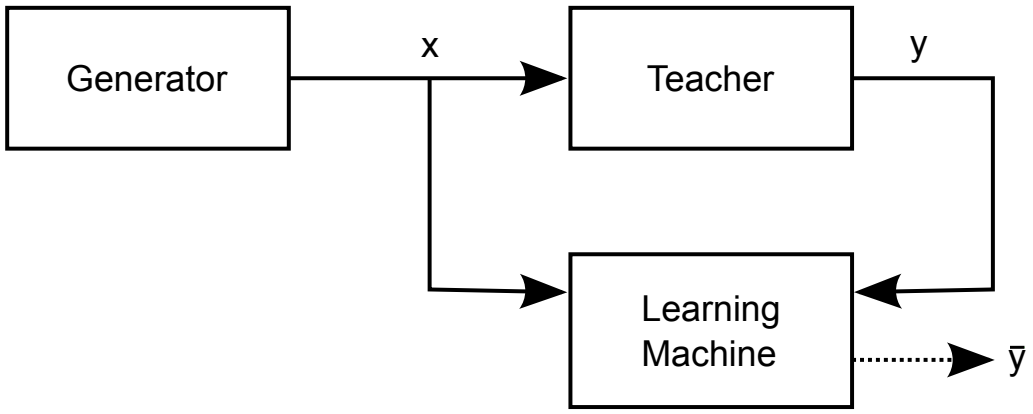


Figure 2.1: Flowchart of supervised learning taken from [22]. The learning machine is trained with pairs (x,y). The input variable x is sampled by a generator process G and labeled by a supervisor T. The aim is to mimic the behaviour of the supervisor as good as possible.

misclassification errors as possible on unseen data. In order to quantify how good a hypothesis performs we measure the expected error further on called the expected risk. For binary classification we can simply use the 0-1 loss-function or Boolean error defined as

$$L_{0-1}(h(x_i), y_i) = \begin{cases} 0 & h(x_i) = y_i \\ 1 & h(x_i) \neq y_i \end{cases} \tag{2.2}$$

As we see later the type of loss-function used during the training affects several aspects e.g. the robustness of the learning machine to noise and the uniqueness of the found solution. Using the 0-1 loss function we define the expected risk which is equal to the probability of misclassification for binary problems

$$\begin{aligned} R(h) = E\left(L_{0-1}\left(h\left(x\right), y\right)\right) &= \int L_{0-1}\left(h\left(x\right), y\right) dP\left(x, y\right) \\ &= \int \int L_{0-1}\left(h\left(x\right), y\right) dP\left(y|x\right) dP\left(x\right) \end{aligned} \tag{2.3}$$

The expected risk is a measure of how likely the misclassification of an input sample is over all possible inputs. Intuitively the expected risk of the classifier should be as low as possible. From this point of view learning in general is a risk minimization problem, the best classifier is selected based on the training data in order to minimize the risk.

There are two different methods to find the appropriate hypothesis h which minimizes the estimated risk. The first one is to imitate the supervisors behavior by

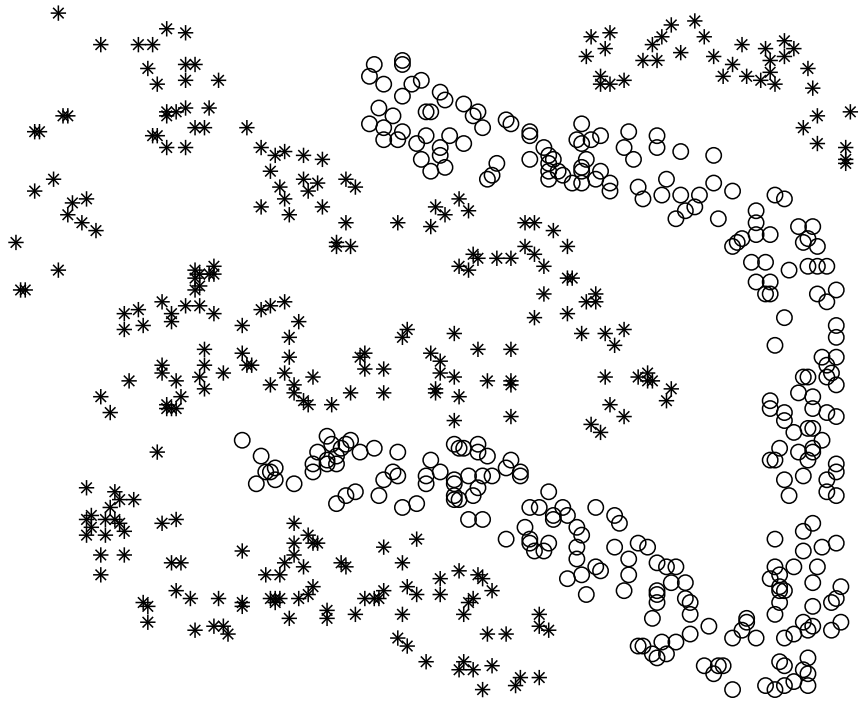


Figure 2.2: A classification problem in two dimensional space which is difficult to solve using generative approaches (UCI Fourclass dataset)

directly finding a mapping from $\mathcal{X} \to \mathcal{Y}$. Learning machines which find this direct mapping are called discriminative methods such as the support vector machine or the perceptron algorithm. The second one is to identify the behavior of the supervisor by modeling the distributions of the input and output variables in order to minimize the expected risk. Those methods are referred to as generative or parametric approaches such as the Naive Bayes classifier. The generative and discriminative approaches differ in complexity and in the kind of prior knowledge which can be incorporated in the classifier design.

Generative Learning algorithms consist of two stages. In the inference stage the class-conditional probabilities $p(x|y)$ as well as $p(y)$ and $p(x)$ are estimated from the training examples. The problem is to find the parameters of a given family of distributions that explain the given observations in the best possible way, one solution is to find the parameters such that the observations becomes most likely. Based on those distributions the posterior class probabilities $p(y|x)$ are determined for each class using the Bayes law. In a second stage decision theory is used in combination with $p(y|x)$ to find the class-membership of the unknown instance x [4]. Vapnik describes the first step as from particular to general (inductive) and the second step as from general to particular (deductive step) [22]. The disadvantage of the generative algorithms is that modeling $p(x|y)$ can be difficult, especially given situations where the instances lie in a high dimensional space and the number of

training samples is small. Figure 2.2 demonstrates a two dimensional classification problem where it is difficult to model $p(x|y)$. The advantages of generative approaches is that abnormal points in the training set can be identified by evaluating $p(x)$.

The discriminative methods find the discriminative function mapping from $\mathcal{X} \rightarrow \mathcal{Y}$ directly based on the training set S, they do not try to model the distributions of the input and output variables. The complexity of the stated problem is lower since only the one direct step must be solved. The advantage is that we do not face the curse of dimensionality by trying to parameterize a distribution in a high-dimensional input space. On the other hand we can not utilize the posterior probabilities $p(y|x)$ in order to determine how confident the classifier is, given a specific value of x.

As mentioned earlier the overall goal is to minimize the expected risk, the problem is that the joint probability distribution is unknown $p(x, y)$. Discriminative approaches solving a classification problem therefore minimize the empirical risk instead R_{emp} of the expected risk defined as

$$R_{emp}(h) = \hat{E}\left(L_{0-1}\left(f(X), Y\right)\right) = \frac{1}{l}\sum_{i=1}^{l} L_{0-1}\left(h(x_i), y_i\right) \ , (x_i, y_i) \in S \tag{2.4}$$

However if the selection of the best h from the class of all possible hypothesis $\mathcal{H}$ is solely based on the empirical error, it is not guaranteed that the expected risk is minimized [20]. To see this imagine a hypothesis which yields a zero training error by remembering all training examples, nothing is learned about the underlying concept of the problem and we risk to fail in classifying unseen examples correctly. This problem is typical for supervised learning and referred to as overfitting. In order to avoid overfitting and enforce a small number expected errors on unseen data, the learning machine must be able to generalize well or in other words to have a small generalization error $|R(h) - R_{emp}(h)|$ [24], again this requirement can not be used directly to select the best hypothesis since the expected risk R is unknown.

To summarize this section, the goal of machine learning is to learn the underlying dependencies between input and output variables as good as possible by generalizing from the training data. In order to learn this dependence empirical risk minimization is applied. The open questions is what are conditions for good generalization. This question was answered by Vapnik within his work on Statistical Learning Theory which led to the development of support vector machines, in the following the main findings of the statistical learning theory are described.

2.2 Statistical Learning Theory

Learning theory is a field of theoretical computer science and analyzes machine learning algorithms. A learning theory is used as a framework to determine which kind of problems are learnable by a given algorithm and how good (convergence rates, bounds on the error) this learning is. Some examples of learning theories are Bayesian leaning, probably approximately correct learning and the statistical learning theory (SLT). The SLT first introduced by Vladimir Vapnik in the late 1960s [22], is a general learning theory which considers classification, regression and density estimation problems as special cases. Vapnik analyzed the principle of inductive inference on a mathematical basis in order to find conditions under which the law of large numbers holds for a hypothesis class. Namely among what conditions the empirical risk of a hypothesis chosen from a hypothesis class converges to the true risk. Based on the empirical risk minimization, that is used by all inductive learning algorithms, he determined conditions under which the learning process is consistent, bounds on generalization and how the generalization rate of a learning machine can be controlled. In his analysis he focused especially on how learning should be performed in the case of small training sample sizes, and models the problem of learning as a direct learning approach. His intention is to solve the learning problem directly, if the amount of information is restricted and to "never solve a more general problem as an intermediate step" [22].

2.2.1 Empirical Risk Minimization

The problem of empirical risk minimization (ERM) is to find the hypothesis h_l from a class of functions $\mathcal{H}$ which minimizes the empirical risk. Statistical learning theory was developed by applying the direct method, therefore the empirical risk minimization is done based on problem formulation in 2.5. Given a collection of training samples S of size l, we determine the best hypothesis h_l by minimizing

$$h_l = \arg\min R_{emp}(h) = \frac{1}{l}\sum_{i=1}^{l} L\left(h\left(x_i\right), y_i\right) \ , (x_i, y_i) \tag{2.5}$$

Different from equation 2.4 the above equation uses a general loss function since the SLT is a general learning theory independent of the specific learning problem solved, it can be applied to classification as well as regression or density estimation.

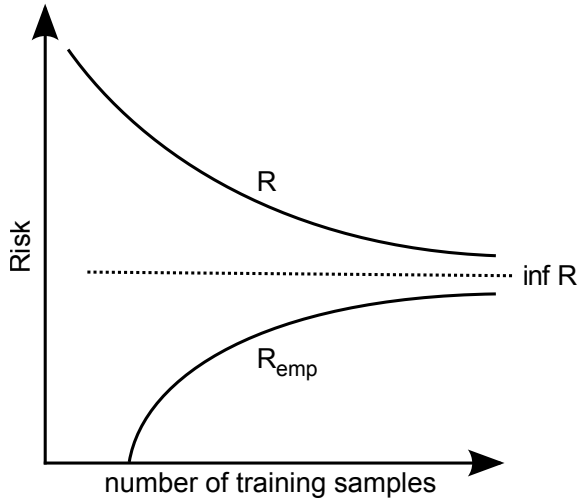


Figure 2.3: Empirical process is consistent if the expected and empirical risk converge to the minimal possible value of the Risk, after [22]

2.2.2 Consistency

Consistency is a general property of a learning process, in order to define it we introduce two classifiers.

Let $h_{\mathcal{H}}$ be the best classifier in $\mathcal{H}$ the one which minimizes the expected risk and let h_l be the classifier selected from $\mathcal{H}$ based on a set of training examples

$$h_{\mathcal{H}} = \arg\min R(h)$$
$$h_l = \arg\min R_{emp}(h)$$

A learning method is consistent if the following two conditions are fulfilled

$$R(h_l) \rightarrow^{P}_{l\rightarrow\inf} R(h_{\mathcal{H}}) \, R_{emp}(h_l) \rightarrow^{P}_{l\rightarrow\inf} R(h_{\mathcal{H}}) \tag{2.6}$$

This means that if the number of training samples is increased to infinity, the expected risk of the classifier produced by the learning machine must converge to the best achievable solution given the function class $\mathcal{H}$. Furthermore the empirical risk of the classifier must converge to this expected risk as shown in figure 2.3. As pointed out by Luxburg et al.[24] consistency is therefore a property of $\mathcal{H}$ and not of a single function h. Vapnik showed that if h_l is determined based on empirical risk minimization the necessary and sufficient condition for consistency is uniform convergence of $R_{emp}(f)$ to $R(f)$ for all possible hypothesis $h \in \mathcal{H}$ as shown in the next equation

$$\lim_{l\rightarrow\infty} P\left(\sup_{f\in\mathcal{F}} |R(f) - R_{emp}(f)| \geq \epsilon\right) \rightarrow 0 \ , \ \forall\epsilon \tag{2.7}$$

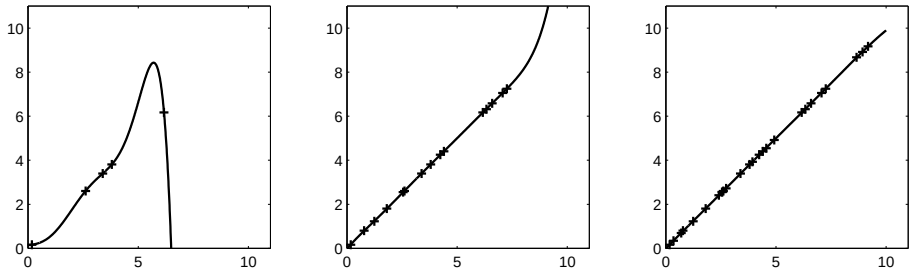


Figure 2.4: An overfitting example

.

This means that for all hypotheses h in hypothesis class of $\mathcal{H}$ it holds that by increasing the amount of training data the empirical risk of the hypothesis approximates the true risk. In that sense the requirement for consistency of ERM is that the law of large numbers takes place in the hypothesis class $\mathcal{H}$ for all h simultaneously since the hypothesis h_l is not fixed due to its dependence on the training set. Thus the consistency, that is success of the empirical risk minimization depends, on the generalization ability of the specific hypothesis class $\mathcal{H}$ used in the learning process.

In order to illustrate a case which violates this condition, let us have a look at the overfitting example of figure 2.4. All three hypotheses yield zero training error $R_{emp}(h) = 0$, but only one of them also minimizes the expected error. In this specific example training sets of sizes of 5, 10 and 15 were used. The hypothesis class is the class of function $\mathcal{H}_7$ of all polynoms with degree ≤ 7. The underlying function where the training instances are sampled from is, $f(x) = x$. We see that the hypothesis learned on empirical risk minimization is not unique and that it yields a well generalizing or overfitting hypothesis. However if the number of training samples is increased, the probability of overfitting decreases.

Let us assume the hypothesis class would be designed in such a way that the maximum degree of the polynoms in $\mathcal{H}$ grows with the number of training samples $\mathcal{H}_{n-1}$. In this case it is always possible to find a hypothesis which interpolates the given training samples perfectly but fails to approximate the true underlying function. The conditions for convergence stated in equation 2.7 would be violated. In the following we show how SLT approaches this problem by introducing a capacity measure for $\mathcal{H}$ and relating it to the generalization ability of h.

2.2.3 Capacity

In order to evaluate which properties $\mathcal{H}$ should have such that the conditions for convergence hold, it is necessary to define bounds on $P\left(\sup_{f\in\mathcal{F}}|R(f) - R_{emp}(f)|\right)$ depending on $\mathcal{H}$. By analyzing those bounds Vapnik proofed that a necessary and

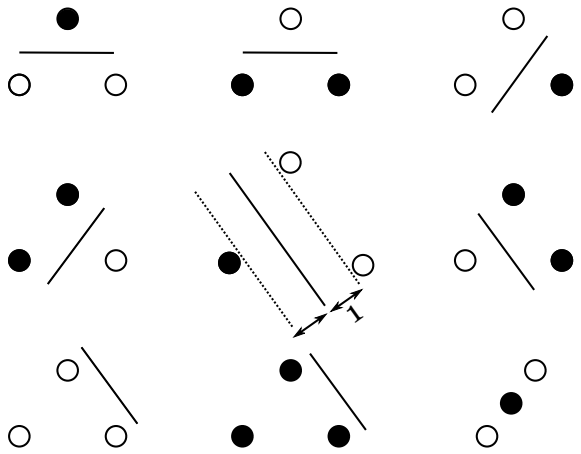


Figure 2.5: In a two dimensional space a maximum of three points can be completely shattered by a plane.

sufficient condition for convergence in the context of classification problems is

$$\lim_{l\to\infty} \frac{(\ln \mathcal{N}(\mathcal{H}, l))}{l} = 0 \ , \ \forall \epsilon > 0 \tag{2.8}$$

The shattering coefficient $\mathcal{N}$ is a capacity measure of $\mathcal{H}$ for classification problems, it defines the maximum number of different separations that can be obtained using the functions of $\mathcal{H}$. Given l instances, the shattering coefficient is at maximum 2^l since every point can be labeled in 2 ways. The meaning of equation 2.8 is, that if $\mathcal{H}$ is so rich, that the number of correct separations grows exponentially in l no learning takes place. In this case a hypothesis h which explains the training set fully without any error can always be found.

Based on the shattering coefficient, Vapnik and Chervonenkis introduced another capacity measure, the VC-Dimension. It is defined as the maximum sample size l which can be completely shattered by $\mathcal{H}$

$$VC(\mathcal{H}) = \max\left(l |\ \mathcal{N}(\mathcal{H}, l) = 2^l\right) \tag{2.9}$$

As we see the VC-dimension is a worst case measure, if h can shatter all points in S than no learning takes place. If all points are shattered by h it is impossible to determine if h is a good model based on the training set. Because it is not possible to find an instance $x_i \in S$ which could show that $h(x_i) \neq y_i$ is a bad model. The VC-dimension is, as well as the shattering coefficient, independent of the underlying distribution of the samples which are shattered. The example in figure 2.5 shows a classifier with VC-Dimension 3 which can shatter three points. A VC-dimension of 3 does however not mean that all sets of size 3 can be shattered, as shown in the example at the bottom right. For the figure in the middle we have shown a special case, the plane separating the two classes of points is situated such that the distance

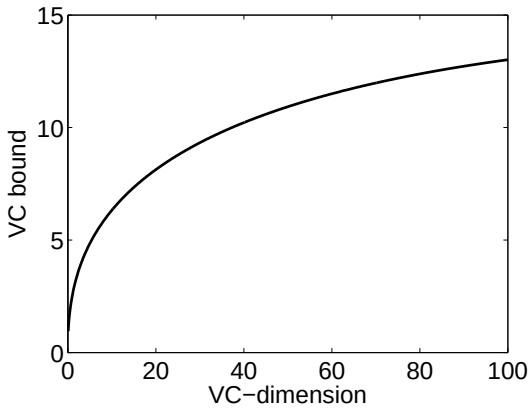


Figure 2.6: The value of the VC bound depending on the VC dimension of $\mathcal{H}$ for $l = 100$ and $\delta = 0.1$.

to the nearest points is equal. This kind of separation is called optimal separating hyperplane and it is the principle the support vector machines are built upon, as we will discuss later. In general the VC-dimension of a separating hyperplane in $\mathbb{R}^N$ upper bounded by $VC(h) = N + 1$ [20], so that the capacity of the classifier depends only on the number of points needed to define the hyperplane.

One is of course not limited to the use of hyperplanes in order to shatter points. Bartlett et al. [2] show for example how to determine the VC-dimension of Neural Nets in order to determine the number of training points needed to achieve a good generalization of the network. It is important to point out that the VC dimension does not necessarily depend on the number of parameters used to specify h, for example the function $h : \mathbb{R} \to \{+1, -1\} = \text{sign}(\sin(\alpha x))$ with one free parameter is able to shatter infinite points as shown in [6]. In addition to the VC-dimension there are other capacity concepts defined in SLT such as VC-entropy, annealed VC-entropy of the growth function which are more specific than the VC-dimension [19] and lead to stricter bounds but are more difficult to determine.

Let us now see how the capacity measure is linked to the generalization ability of a specific hypothesis. The expected risk of a hypothesis h which minimizes the empirical risk given an arbitrary training set of l samples satisfies

$$R(h_l) \leq R_{emp}(h_l) + \underbrace{\sqrt{\frac{1}{l}\left(d\left(\ln\frac{2l}{d}+1\right)+\ln\frac{4}{\delta}\right)}}_{\text{capacity term}} \tag{2.10}$$

with probability of at least $1 - \delta$ given the VC-dimension $d = VC(\mathcal{H})$ [20]. This equation allows to restrict the hypothesis class $\mathcal{H}$ in a constructive way so as to

control the generalization ability of the learning machine. The term depending on d is also called capacity term as referred to by [20],[24].

2.2.4 Structural Risk Minimization

In order to complete our discussion about SLT, let us now see how to select $\mathcal{H}$ such that bound on the expected risk is minimized. The idea of structural risk minimization (SRM) is to reduce the bound defined by 2.10 by minimizing the empirical error and the capacity term simultaneously. Vapnik proposes to use a hypothesis class $\mathcal{H} = \mathcal{H}_1 \subset \mathcal{H}_2 \ldots \subset \mathcal{H}_n$ which is composed of nested subsets with finite VC-dimension and choose the h_l from the set $\mathcal{H}$ which minimizes 2.10. Referring to the regression example shown in figure 2.4 this means to interpolate with polynomials of different degree and choose the h such that the approximation error as well as the capacity term is minimized simultaneously. SRM is useful especially in cases where the number of training samples is small and the empirical risk R_{emp} is not close to the actual risk R as pointed out by Vapnik in [23]. According to Hastie et al. [15] the main challenge in order to apply SRM is to determine the VC-dimension $\mathcal{H}$ accurately, they state that it is usually only possible to obtain upper bounds on the VC-dimension. Burges noted in [6] that the VC bounds for a support vector machine are very loose depending on the used kernel, but are predictive enough to choose a specific classifier h.

2.3 Optimization Theory

As we have seen in the last section solving the learning problem using empirical risk minimization is an optimization problem. Before we continue and show how this problem is solved using the support vector machine approach, we introduce a mathematical optimization method. The method of Lagrange multipliers describes how a function can be minimized subject to a number of constraints. In our description we follow the definitions given by Boyd in [5]. The optimization problem, also referred to as the primal problem, is solved by Lagrange multipliers is defined as

$$\begin{aligned} \arg\min \quad & f_0(x) \\ \text{subject to} \quad & f_i \leq 0 \; i = 1 \ldots M \\ & h_i = 0 \; i = 1 \ldots P \end{aligned} \tag{2.11}$$

By introducing the Lagrange multipliers α and β the inequality and equality constraints are included in the optimization problem leading to the Lagrangian $\mathcal{L}$

$$\mathcal{L}(x, \alpha, \beta) = f_0(x) + \sum_{i=1}^{M} \alpha_i f_i(x) + \sum_{i=1}^{M} \beta_i h_i(x) \tag{2.12}$$

We assume that the Domain $\mathcal{D}$ of the problem is nonempty, such that an optimal value p^* can be found.

2.3.1 Lagrangian Duality

Let us now see how the optimal value of p^* can be found. For every Lagrangian $\mathcal{L}$ we can define a dual function

$$g(\alpha, \beta) = \min_x \mathcal{L}(x, \alpha, \beta) \tag{2.13}$$

As shown by Boyd the dual function is a lower bound on the optimal solution p^* and the following holds for every feasible point $\tilde{x}$

$$g(\alpha, \beta) = \min_x \mathcal{L}(x, \alpha, \beta) \leq \mathcal{L}(\tilde{x}, \alpha, \beta) \leq f_0(\tilde{x}) \tag{2.14}$$

In order find a nontrivial solution it is necessary to restrict $\alpha \geq 0$ otherwise we could choose the Lagrange multiplier such that $g(\alpha, \beta) = -\inf$ at each feasible point. The Lagrange multiplier β does not need to be positive, since the value of $h_i(\tilde{x})$ for a feasible point is zero as given by the constraint. In order to find the best value of the dual function, that is the lower bound which is the closest to p^*, the following optimization problem must therefore be solved

$$\begin{aligned} \max_{\alpha,\,\beta} \min_x \quad & \mathcal{L}(x, \alpha, \beta) \\ \text{subject to} \quad & \alpha \geq 0 \end{aligned} \tag{2.15}$$

This problem is referred to as the Lagrange dual problem. Let us now state the conditions under which the optimal solution d^* to the dual problem is equal to the optimal solution p^* of the primal problem. If the condition $d^* = p^*$ hold, then the bound obtained from the Lagrangian dual is tight [5] and we speak of strong duality. One constraint under which strict duality holds is Slaters condition[5]. Given that the primal problem of the form

$$\begin{aligned} \min_x \quad & f_0(x) \\ \text{subject to} \quad & f_i \leq 0,\ i = 1 \ldots M \\ & Ax = b \end{aligned} \tag{2.16}$$

it states that there must exist an x in the relative interior of $\mathcal{D}$ such that the inequality conditions f_i are strictly satisfied. In other words for all inequality con-

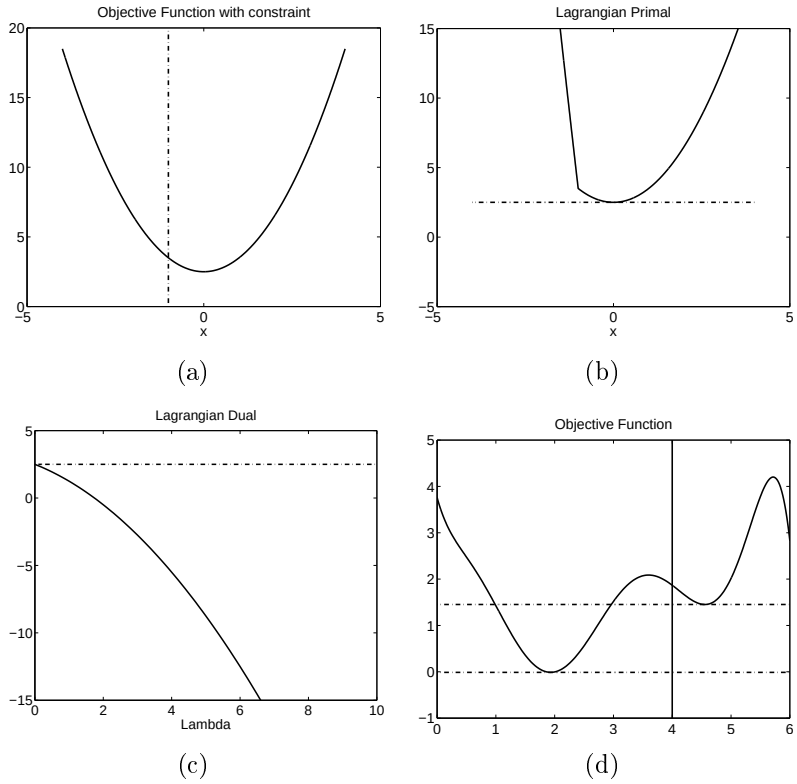


Figure 2.7: The primal and dual solution for a convex (a) and a non-convex optimization problem (d)

straints f_i the following holds $f_i(x) < 0$. If we further restrict the problem assuming that h_i is affine and f_i and f_0 are convex, the Karush-Kuhn-Tucker conditions (KKT) are sufficient for strong duality. For a convex optimization problem the KKT conditions are defined as

$$\begin{aligned} f_i(\tilde{x}) &\leq 0 && \text{primal constraint, } i = 1 \dots M \\ h_i(\tilde{x}) &= 0 && \text{primal constraint, } i = 1 \dots P \\ \tilde{\alpha}_i &\geq 0 && \text{dual constraint} \\ \tilde{\alpha}_i f_i(\tilde{x}) &= 0 && \text{complementary slackness condition} \\ \nabla_x f_0(\tilde{x}) + \sum_i \tilde{\alpha}_i \nabla_x f_i(\tilde{x}) + \sum_i \tilde{\beta}_i \nabla_x h_i(\tilde{x}) &= 0 && \text{first order derivatives of the primal must vanish for } \tilde{x} \end{aligned} \tag{2.17}$$

If the KKT conditions hold for at a point $\tilde{x}$, $\tilde{\alpha}$, $\tilde{\beta}$ then this point is primal and dual optimal.

The equivalence of the primal and dual solution is shown in figure 2.7 (a) - (c). We see how the minimum of the function $f(x) = x^2 + 2.5$ subject to the constraint $x > -1$ is found. The minimum of the Lagrangian primal is found at $p^* = 2.5$ and

the maximum of the dual is also $d^* = 2.5$, both values are equal since the constraint function and the objective function are convex. In figure 2.7(d) a non-convex function f_0 is minimized subject to the convex constraint $-x <= 4$, the horizontal lines indicate p^* and d^*. Using the dual formulation in order to solve this problem we fail to find the correct minimum subject to the constraint leading to a duality gap between the primal and the dual solution.

In this section we have shown how optimization problems can be solved subject to a number of constraints. We have seen that for a particular type of optimization problem, namely a convex problem with convex inequality and affine equality constraints, it is possible to solve the Lagrangian dual (2.15) and obtain the same solution as if the Lagrangian primal (2.14) is solved. Up until now there is no reason why we should prefer to solve the optimization problem in the dual form, but as we will see shortly the dual formulation and especially the complementary slackness condition of KKT have very interesting consequences.

2.4 Support Vector Machines

The former sections described the Statistical Learning Theory and a mathematical method to solve a specific kind of optimization problem, this section shows how those theoretical concepts are applied in order to build a well generalizing non-probabilistic classifier. This algorithm is the support vector machine, it is based on the minimization capacity term with respect to the empirical risk in order to minimize the expected risk.

2.4.1 Hard-margin SVM

The hard margin support vector machine, is a classifier which assigns the label y_i to $\mathbf{x}_i$ based on the distance $f(\mathbf{x}_i)$ of the instance to a decision boundary. The simplest way to separate points is by a hyperplane, by classifying an instance $\mathbf{x}$ according to its distance from the hyperplane defined by vector $\mathbf{w}$ and offset b

$$h(\mathbf{x}, \mathbf{w}, b) = \theta\left(\mathbf{w}^T\mathbf{x} + b\right). \tag{2.18}$$

That function θ represents an indicator function whether or not the distance is positive or negative. In order to find a solution to the above problem, that is to find $\mathbf{w}$ and b which define the hyperplane, the training set S needs to be linearly separable. That is all points of a class are situated on the same side of the hyperplane, satisfying the following inequality

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1,\ \mathbf{x}_i \in \mathbb{R}^N,\ y_i \in \{1, -1\}$$

The hyperplane defined by $\mathbf{w}$ is referred to as a γ-margin separating hyperplane if the following holds for all training examples

$$y_i \frac{\left(\mathbf{w}^T\mathbf{x} + b\right)}{||\mathbf{w}||} \geq \gamma \tag{2.19}$$

As we have see in equation 2.10 from section 2.2 the expected risk is bounded by the empirical risk and the capacity of the the hypothesis class $\mathcal{H}$. In case of the problem of finding the optimal separating hyperplane for a linear separable problem, minimizing the expected risk 2.10 reduces to the minimization of the capacity term. This is because the empirical risk of misclassifying a linearly separable dataset using a separating hyperplane is zero. As already stated the VC-dimension of a hyperplane is independent of the number of training points l and only influenced by the dimensionality of $\mathbf{x}$. We stated previously that the VC dimension is in that case $N + 1$ if $\mathbf{x} \in \mathbb{R}^N$. This bound can be refined as shown by Vapnik [23]. Given training instances from an N-dimensional $\mathbf{x}$ and belonging to a sphere of radius R the VC-dimension of this γ-margin separating hyperplane is

$$VC\left(h\left(\mathbf{x}, \mathbf{w}, b\right)\right) = \min\left(\frac{R^2}{\gamma^2}, d\right) + 1 \tag{2.20}$$

Following the principles of SRM the optimal separating hyperplane from the hyperplane which separates the training set without error, is the one which has the smallest VC-dimension. Or in other words the hyperplane which separates both classes in such a way that the distance from the instances to the hyperplane is maximized simultaneously for both classes as shown in figure 2.8. From the above equation we see that this is condition is fulfilled for the separating hyperplane with the largest geometrical margin γ such that all points are classified correctly. According to the SLT the VC-dimension is defined as a capacity measure for a function class $\mathcal{H}$ with the definition of the VC-dimension given in equation 2.20 it is now possible to evaluate the capacity for a specific function h. That is finding the optimal separating hyperplane corresponds to the following optimization problem

$$\begin{aligned} h\left(\mathbf{x}, \mathbf{w}, b\right)^* = \quad & \max_{w,b} \gamma \\ \text{subject to} \quad & y_i \frac{\left(\mathbf{w}^T\mathbf{x}_i + b\right)}{||\mathbf{w}||} \geq \gamma \ , \ i = 1 \ldots M \end{aligned}$$

Let us now express the geometrical margin γ in terms of the functional margin $\hat{\gamma} =; y\left(\mathbf{w}^T\mathbf{x} + b\right)$

$$\begin{aligned} h\left(\mathbf{x}, \mathbf{w}, b\right)^* = \quad & \max_{w,b} \frac{\hat{\gamma}}{||\mathbf{w}||} \\ \text{subject to} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq \hat{\gamma} \ , \ i = 1 \ldots M \end{aligned}$$

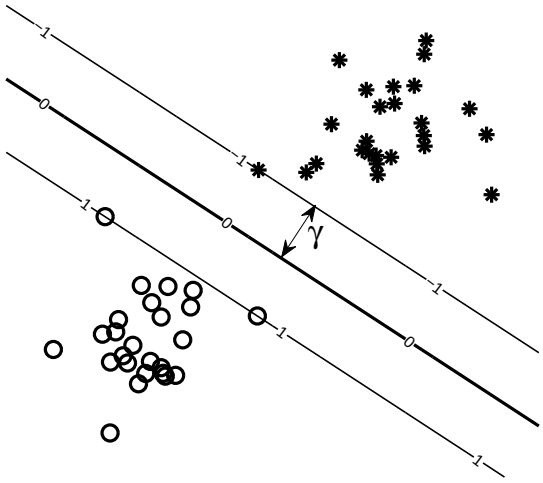


Figure 2.8: The optimal separating hyperplane

Using the fact that we can always scale $\mathbf{w}$ and b by a constant without changing the geometrical margin, we can replace $\hat{\gamma}$ by 1. The optimization problem now simplifies to

$$\begin{aligned} h\left(\mathbf{x}, \mathbf{w}, b\right)^* = & \max_{w,b} \frac{1}{||\mathbf{w}||} \\ \text{subject to} \quad & y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 \ , \ i = 1 \ldots M \end{aligned}$$

which is equivalent to the following quadratic optimization problem [23]

$$\begin{aligned} h\left(\mathbf{x}, \mathbf{w}, b\right)^* = & \min_{w,b} \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 \ , \ i = 1 \ldots M \end{aligned} \tag{2.21}$$

In order to solve this convex quadratic optimization problem given the inequality constraint $y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1$, the method of Lagrange multipliers is used. The Lagrangian of equation 2.21 is defined as

$$\begin{aligned} \mathcal{L}_P\left(\mathbf{w}, b, \boldsymbol{\alpha}\right) = & \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_i \alpha_i \left[y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - 1\right] \\ \text{subject to} \quad & \alpha_i \geq 0 \end{aligned}$$

The optimal separating hyperplane or in other words the hypothesis h^* with the smallest bound on the expected error is found by minimizing the Lagrangian with respect to w, b and maximizing it with respect to α.

$$\begin{aligned} h\left(\mathbf{x}, \mathbf{w}, b\right)^* = & \min_{w,b} \max_{\alpha} \mathcal{L}_P\left(\mathbf{w}, b, \boldsymbol{\alpha}\right) \\ \text{subject to} \quad & \alpha_i \geq 0 \end{aligned} \tag{2.22}$$

This problem is the primal optimization problem, as we can see it corresponds to solving the empirical risk minimization problem regularized by the capacity of the separating hyperplane. Since both the primal problem of minimizing $\frac{1}{2}||\mathbf{w}||^2$ and the constraints are convex, we can equally solve the corresponding dual problem

$$\begin{aligned} h\left(\mathbf{x}, \mathbf{w}, b\right)^* = & \max_{\alpha} \min_{w,b} \mathcal{L}_P\left(\mathbf{w}, b, \boldsymbol{\alpha}\right) \\ \text{subject to} \quad & \alpha_i \geq 0 \end{aligned} \tag{2.23}$$

In order to minimize the dual with respect to $\mathbf{w}, b$ let us take the derivative and set it to zero

$$\begin{aligned} \frac{L_P}{\partial \mathbf{w}} &= \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i \\ \mathbf{w} &= \sum_i \alpha_i y_i \mathbf{x}_i \\ \frac{L_P}{\partial b} &= -\sum_i \alpha_i y_i \\ 0 &= \sum_i \alpha_i y_i \end{aligned} \tag{2.24}$$

Substituting the above results into equation 2.23 we obtain the so called Wolfe dual.

$$\begin{aligned} L_D = & \max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & \sum_i \alpha_i y_i = 0,\ \alpha_i > 0 \end{aligned} \tag{2.25}$$

After finding the values for α_i^* by applying an algorithm to solve this problem, we need to determine the optimal value for the offset b since it is excluded from the dual solution. We know that the KKT conditions need to be satisfied in order to make α_i^* optimal in the sense of zero duality gap, therefore we can exploit the complementary slackness conditions. We find b such that $\alpha_i \left(y_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) - 1\right) = 0$ is satisfied for all x_i with $\alpha_i = 0$. In order to ensure numerical stability, Burges([6]) points out to take the mean value of all b resulting from the single equations. Once the optimal separating hyperplane is defined by determining $\mathbf{w}$ and b we are able to classify a previously unseen instance $\mathbf{x}$ by evaluating $\mathbf{w}^T \mathbf{x} + b$ in the following way

$$\begin{aligned} h(\mathbf{x}, \mathbf{w}, b) &= \operatorname{sgn}\left(\mathbf{w}^T\mathbf{x} + b\right) \\ &= \operatorname{sgn}\left(\left(\sum_i \alpha_i y_i \mathbf{x}_i\right)^T \mathbf{x} + b\right) \\ &= \operatorname{sgn}\left(\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b\right) \end{aligned} \tag{2.26}$$

Let us now characterize the properties of the solution. The classification of $\mathbf{x}$ depends on the linear expansion of $\mathbf{w}$ in terms of the training points $\mathbf{x}_i$. This expansion is sparse, this follows from the complementary slackness condition. Only those points where $y_i\mathbf{x}_i^T\mathbf{w} + b = 1$ have a weight $\alpha_i > 0$, since the value of α_i can be maximized without violating the complementary slackness condition. Those points are referred to as support vectors and the algorithm solving for the optimal separating hyperplane by maximizing the margin is called hard-margin support vector machine. In figure 2.8 we see that only three points have a functional margin of 1, those points define the optimal separating hyperplane. All other points of the training set are not taken into account when defining the decision boundary.

The advantage of the hard-margin support vector machine over other classifiers such as Neural Networks is that the optimization problem has a unique solution, the learning process is rather fast and by constructing the decision rule one obtains a set of support vectors [23]. We refer to unique solution in the sense, that there is one hyperplane which separates the two classes in the best possible way and that any solution found for equation 2.23 is a global minimum. Furthermore using the SVM allows for training with small sample sizes in cases where the empirical risk minimization does not guarantee a small value of the expected risk R, due to the fact that the optimization objective is to minimize the capacity term.

The disadvantage of the hard-margin support vector machine is that it is defined for linearly separable datasets, which is a strong assumption and limits the applicability of the classifier to real-world data sets. There are two principle ways to deal with the problem of linearly non-separable datasets, either to map the data into a higher dimension where the data is linear separable or to formulate the problem so that a certain amount of misclassification is tolerated. The former approach is called the kernel trick and leads to a non-linear decision surface in the feature space, the later one is the soft-margin support vector machine, both methods are usually combined. We will discuss both of them in the following sections.

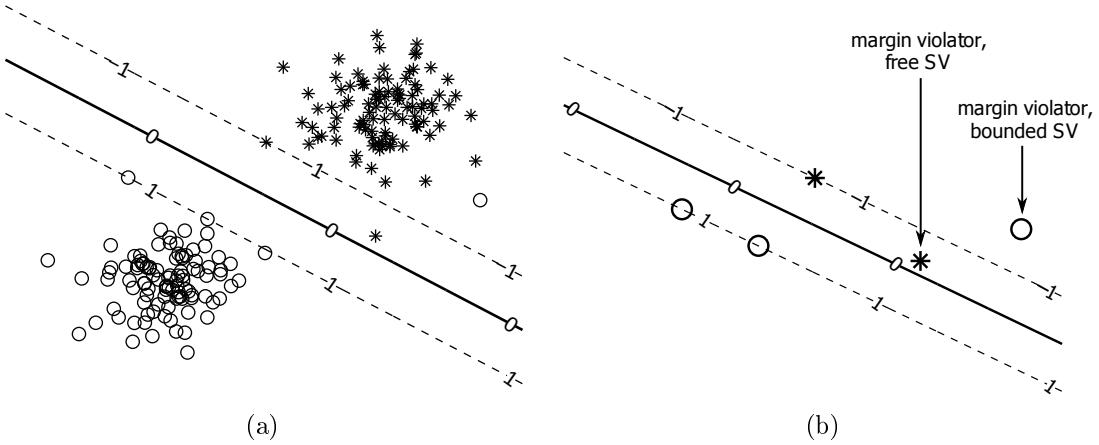


Figure 2.9: Soft-margin SVM using a linear kernel, (b) shows the support vectors which define the hyperplane

2.4.2 Soft-margin SVM

Since linear separability of two classes can not always be assumed, slack variables ξ_i were introduced by Cortes and Vapnik [11] in order to relax the separability condition of the hard-margin SVM. A value of $\xi > 0$ is assigned to all points that are situated either on the wrong side of the hyperplane or inside the functional margin $\mathbf{w}^T\mathbf{x} + b \leq 1$. Consequently the optimization problem in equation 2.21 is relaxed in such a way that margin violation is accepted $\phi(\xi) = \sum_{i=1}^{M} \xi_i$. All points for which $0 \leq \xi_i \leq 1$ holds, are situated on the correct side of the hyperplane, whereas all $\xi_i > 1$ are misclassified since they lie on the wrong side of the hyperplane. Additionally to the slack variable the cost parameter C is introduced in order to balance the two optimization objectives to minimize $\mathbf{w}^T\mathbf{w}$ and to ensure that the number of margin violators is small.

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i \\ \text{subject to} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{2.27}$$

This constrained convex quadratic optimization problem can also be expressed in an unconstrained way introducing the hinge loss function

$$\begin{aligned} L_{hinge}\left(y_i, \mathbf{w}^T\mathbf{x}_i + b\right) &= \max\left(0, 1 - y_i, \mathbf{w}^T\mathbf{x}_i + b\right) \\ \min_{w} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i L_{hinge}\left(y_i, \mathbf{w}^T\mathbf{x}_i + b\right) \end{aligned} \tag{2.28}$$

The loss function allows us to easily examine how margin violators contribute to the minimization problem and is a way to increase the robustness of the support

vector machine, as we will see later. This unconstrained optimization problem can be used by applying quadratic solvers or Newton type algorithms. However, the standard formulation of the soft-margin SVM algorithm uses the Lagrange multiplier method to solve the constrained optimization problem in the following way.

The Lagrangian of equation is defined as

$$\begin{aligned} L_P &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i - \sum_i \alpha_i \left(y_i\left(\mathbf{x}^T\mathbf{w} + b\right) + \xi_i - 1\right) - \sum_i \mu_i\xi_i \\ \text{subject to} \quad & \alpha_i \geq 0\ ,\ \mu_i \geq 0 \end{aligned} \tag{2.29}$$

In order to maximize it with respect to α_i set its derivatives with respect to $\mathbf{w}$, b and ξ_i to zero

$$\begin{aligned} \frac{L_P}{\partial \mathbf{w}} &= \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i \\ \frac{L_P}{\partial b} &= -\sum_i \alpha_i y_i \\ \frac{L_P}{\partial \xi_i} &= C - \alpha_i - \mu_i \\ \mathbf{w}_0 &= \sum_i \alpha_i y_i \mathbf{x}_i \\ 0 &= \sum_i \alpha_i y_i \\ 0 \leq \alpha_i &\leq C \end{aligned} \tag{2.30}$$

Finally we find the corresponding dual as

$$\begin{aligned} L_D &= \max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \end{aligned} \tag{2.31}$$

Obviously equation 3.13 is very similar to the solution we found for the hard-margin SVM 2.23 except that the Lagrange multiplier α_i is not unbounded anymore, but upper bounded by C. It must satisfy the box constraints $0 \leq \alpha \leq C$. Therefore we can distinguish between two different kinds of support vectors, all points which satisfy $0 < \alpha_i < C$ are free support vectors, those points are margin violators with $0 < \xi < 1$ which are situated at the correct side of the hyperplane. All points which are situated on the wrong side of the hyperplane are characterized by $\xi > 1$ and $\alpha_i = C$, those points are also called bounded support vectors. This type of support vector machine is called L1 soft-margin SVM.

2.4.3 Nonlinear SVM

As we have seen in the last section one approach to deal with data which is not linearly separable is to tolerate a certain amount of misclassification. However the relaxation of the constraint in equation 2.21 does not decrease the bound on the expected risk. The idea of the nonlinear SVM is to map the data from the original feature space $x \in \mathcal{X}$ to a higher dimensional space $\mathcal{Z}$ where it its linearly separable, or where at least the margin is increased. The linear decision boundary in this high dimensional space corresponds to a nonlinear decision boundary in the original space. In that sense we incorporate an inductive bias, the knowledge that the data is linearly separable in the high dimensional space, in order to solve the classification problem. Let $\phi : \mathcal{X} \to \mathcal{Z}$ be such a mapping, and let $\mathcal{Z}$ be a space which is equipped with an inner product, then we can rewrite the Lagrangian dual 3.13 as

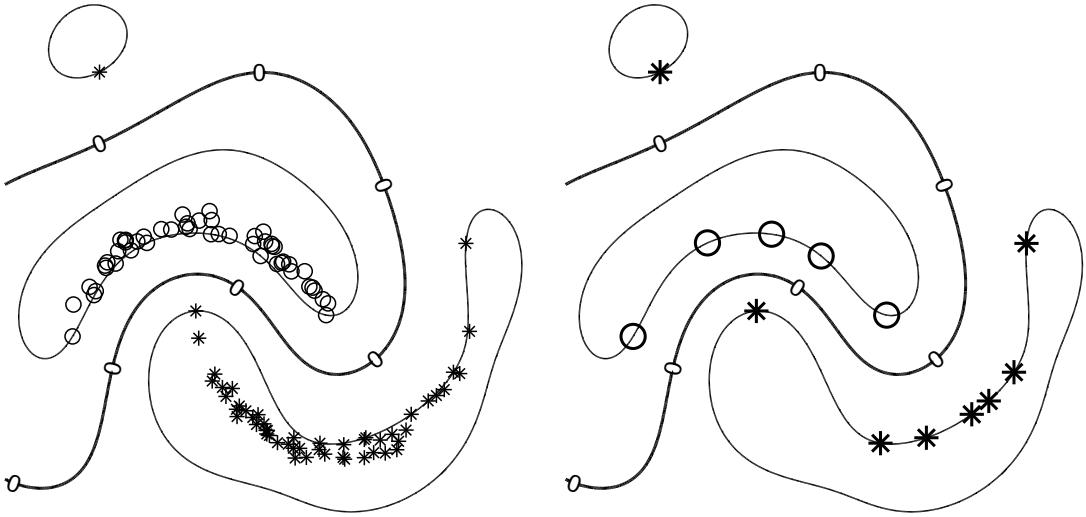
$$\begin{aligned} L_D \; &= \max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi\left(\mathbf{x}_i\right)^T \phi\left(\mathbf{x}_j\right) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \end{aligned} \tag{2.32}$$

However solving would be computationally complex, since it requires to evaluate the inner product in the high-dimensional space $\mathcal{Z}$. Therefore kernels are introduced. A kernel is a function which evaluates the inner product $\phi\left(\mathbf{x}_i\right)^T \phi\left(\mathbf{x}_j\right)$ in the original feature space $\mathcal{X}$ such that

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \phi\left(\mathbf{x}_i\right)^T \phi\left(\mathbf{x}_j\right) \tag{2.33}$$

The kernel can be interpreted as a function which measures the similarity measure between two points $\mathbf{x}_i, \mathbf{x}_j$. Therefore the usual approach is to define a similarity function K for a given problem such that it fulfills the properties of a kernel and not to define a high dimensional feature space and determine the kernel function afterwards. One criterion in order for a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ to be a kernel is that all finite kernel matrices must be positive semidefinite. A good overview how to select, construct kernels and combine kernels can be found in [12].

In the following some popular kernels functions are listed, apart from the linear classifier which uses no kernel, the radial basis function (RBF) kernel is the most used kernel in practice [21]. Note that the RBF kernel uses the same function as the Parzen based classifier, however a Parzen window estimator does not yield the same results since it evaluates the kernel at all training points different from the SVM which only evaluates the kernel values of the support vectors.



(a) (b)

Figure 2.10: Non-linear soft-margin SVM using a RBF kernel, (b) shows the support vectors which define the hyperplane

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= \langle x_i, \mathbf{x}_j \rangle = \mathbf{x}_i^T \mathbf{x}_j && \text{Linear Kernel} \\ K(\mathbf{x}_i, \mathbf{x}_j) &= (\langle x_i, \mathbf{x}_j \rangle + 1)^p && \text{Polynomial Kernel} \\ K(\mathbf{x}_i, \mathbf{x}_j) &= \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{2\sigma^2}\right) && \text{RBF Kernel} \end{aligned} \tag{2.34}$$

Figure 2.10 shows the nonlinear decision boundary of a SVM induced by using the RBF kernel. The RBF kernel evaluates the Euclidean distance from a point to a support vectors due to this the shape of the decision boundary at a single point is circular or elliptic, as visualized at the top of the figure. By weighting with α_i and linearly combining the response of the kernel for all support vectors, the nonlinear decision boundary is formed.

2.4.4 Robustness of the SVM

The robustness of support vector machines was studied by Steinwart and Christmann in [21] and more general with regard to the robustness properties of convex minimization methods in [9]. More recently Hable and Christmann [14] studied the qualitative robustness of support vector machines. Xu et al. [27] also studied the robustness properties of the SVM and showed that the standard support vector machine solution is equivalent to the solution of a robust optimization problem.

Steinwart applies methods of robust statistics to derive conditions under which non-linear soft-margin support vector machines are robust. He analyzes how changes in the distribution of the random variables X and Y affect the classifier's accuracy depending on the loss function, kernel used and cost parameter C. Steinwart uses

the influence function from robust statistics to assess the robustness of support vector machines. An influence function measures the impact of a small amount of contamination of the original distribution P of X and Y in the direction of a point z on the quantity of interest [21]. In other words it evaluates how the accuracy of the classifier changes if the data used for training is contamined by noise. Based on the theoretical analysis Steinwart concludes that if the first derivative of a convex margin-based loss function is bounded and a bounded continuous kernel is used, then the influence function of the SVM is bounded. That is the standard SVM using the hinge loss function in combination with a bounded kernel, as for example the RBF kernel, has a good statistical robustness property. Furthermore Steinwart shows that the robustness of a SVM can be increased by choosing a small value of C.

3. ROBUST SVM APPROACHES

After introducing the theoretical background of support vector machines in the last chapter, we now discuss the different approaches that have been proposed to improve the robustness of the SVM algorithm against outliers by using loss functions different than the hinge loss. The aim of using alternative loss functions is to make the classifier robust against outliers, different from those approaches who filter the training data before training the final classifier. The approach of making the classifier robust against outliers is attractive because explicit filtering is usually computationally complex since it is either based on the estimation of the data distribution or on the construction of new features. An example of a robust SVM formulation which is based on explicit filtering is the weighted support vector machine proposed by Yang et al. [29]. They propose to apply the kernel-based possible c-means algorithm as a preprocessing step in order to weight each training instance according to its euclidean distance from the cluster centers. In the following we present the different approaches which rely on implicit outlier filtering by using a modified loss function.

3.1 The Influence of the Loss Function on the SVM

Before we start and show what kind of different loss functions for SVMs have been proposed and how the optimization problems are solved, we first show how the loss function influences the decision boundary following the discussion of Chapelle [8]. The unconstrained optimization problem of a soft-margin support vector machine with an arbitrary loss function is given as

$$\min_{w} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i L\left(y_i, \mathbf{w}^T\mathbf{x}_i + b\right). \tag{3.1}$$

If we drop the offset term b and set $f(\mathbf{x_i}) = \mathbf{w}^T\mathbf{x}_i$, we can simplify rewrite the problem as

$$\min_{w} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i L\left(y_i, f(\mathbf{x_i})\right). \tag{3.2}$$

We differentiate the above equation with respect to $\mathbf{w}$ and set the derivative to zero in order to minimize the optimization problem

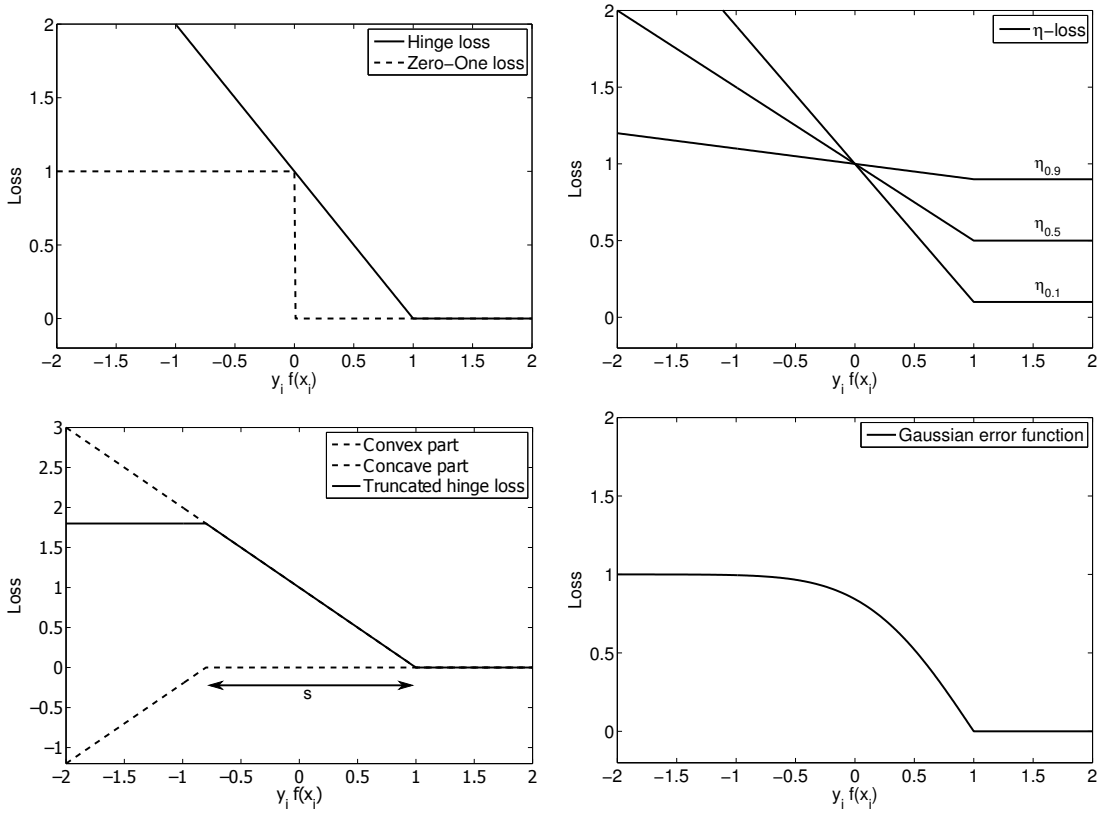


Figure 3.1: Four different loss functions

$$\begin{aligned} \mathbf{w} + C \sum_i \frac{\partial L\left(y_i, f\left(\mathbf{x_i}\right)\right)}{\partial \mathbf{w}} &= 0 \\ \mathbf{w} + C \sum_i \frac{\partial L\left(y_i, f\left(\mathbf{x_i}\right)\right)}{\partial f\left(\mathbf{x_i}\right)} \frac{f\left(\mathbf{x_i}\right)}{\partial \mathbf{w}} &= 0 \\ \mathbf{w} + C \sum_i \underbrace{\frac{\partial L\left(y_i, f\left(\mathbf{x_i}\right)\right)}{\partial f\left(\mathbf{x_i}\right)}}_{\beta_i} \mathbf{x_i} &= 0. \end{aligned} \tag{3.3}$$

From the above equation we see that $\mathbf{w}$ can be written as a linear combination of training instances $\mathbf{x}_i$. Equation 3.3 directly shows that the influence of a training instance $\mathbf{x}_i$ on the decision boundary defined by $\mathbf{w}$ depends on the loss function used. Each training instance $\mathbf{x}_i$ is weighted by the derivative of the loss function with respect to $f\left(\mathbf{x}_i\right)$. Thus instances which are located in a flat area of L have no influence on the expansion of $\mathbf{w}$. This is because including them in the expansion would not reduce the objective value of equation 3.1.

The hinge loss used in the standard SVM formulation is linear, that means all margin violators with $y_i f\left(\mathbf{x}_i\right) < 1$ contribute with the same weight C to $\mathbf{w}$. This

means the larger the loss of $\mathbf{x_i}$ the more beneficial is it to include the point in the expansion of $\mathbf{w}$. This is indeed the same result as we achieved by solving the soft-margin support vector machine in the dual. From the complementary slackness condition we know, that only the margin violator contribute with $0 < y_i\alpha_i < C$ to the expansion of $\mathbf{w}$.
Therefore the aim of the robust SVM methods is to use loss functions which have flat regions for points whose distance from the decision boundary $y_i f(\mathbf{x}_i)$ is very large and can be regarded as outliers. Effectively the loss functions are non-convex which makes it necessary to use other optimization methods than in the standard SVM formulation. In the following we will discuss the robust SVM approaches based on the loss functions shown in figure 3.1.

3.2 η-Hinge Loss Function

Along with a filtering approach Xu et. al [28] present the η-hinge loss. They bound the influence on outliers by minimizing the SVM optimization problem over a set of loss functions. The corresponding robust eta loss function is defined as

$$L_\eta(y_i, f(\mathbf{x}_i)) = \eta_i \left[(1 - y_i f(\mathbf{x}_i) + b)\right]_+ + 1 - \eta_i \tag{3.4}$$

with $f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + b$. The value of η_i is bounded to by $0 \leq \eta_i \leq 1$ and all training instances $\mathbf{x}_i$ which can be regarded as outliers will be assigned a value of $\eta_i = 1$. All other instances are assigned $\eta_i = 0$. The effect of setting η_i to 1 for all outliers is, that the corresponding loss function will be a constant. All instances $\mathbf{x}_i$ with $L_\eta(y_i, f(\mathbf{x}_i))$ inside a constant region are filtered and will not become support vectors. Therefore the optimization objective of the η-hinge loss is to minimize η_i. The primal form of the optimization problem for the η-hinge loss is stated as

$$\begin{aligned} \min_{\mathbf{w}} \quad \min_{\boldsymbol{\eta}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \eta\left[(1 - y_i f(\mathbf{x}_i) + b)\right]_+ + 1 - \eta \\ \text{subject to} \quad & 0 \leq \eta_i \leq 1 \end{aligned} \tag{3.5}$$

As pointed out by Xu minimizing the above equation by alternating the minimization of $\mathbf{w}$ and $\boldsymbol{\eta}$ yields boolean solutions with $\eta_i = 0$ for all outliers and $\eta_i = 1$. They state that this approach is prone to yield solutions which are local minima. Therefore they reformulate the problem in order to simultaneously optimize $\mathbf{w}$ and $\boldsymbol{\eta}$ by relaxing the dual formulation of the problem in the following way:

$$
\begin{aligned}
&\min_{0\leq\boldsymbol{\eta}\leq 1} \min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \eta_i \left[1 - y_i {\mathbf{x_i}}^T \mathbf{w}\right]_+ + 1 - \eta_i \\
&= \min_{0\leq\boldsymbol{\eta}\leq 1} \max_{0\leq\boldsymbol{\alpha}\leq C} \boldsymbol{\eta}^T(\boldsymbol{\alpha} - \mathbf{e}) - \frac{1}{2}\boldsymbol{\alpha}^T\left(\mathbf{Y}\mathbf{X}^T\mathbf{X}\mathbf{Y} \circ \boldsymbol{\eta}\boldsymbol{\eta}^T\right)\boldsymbol{\alpha} + t \\
&\quad \text{with } G = \mathbf{Y}\mathbf{X}^T\mathbf{X}\mathbf{Y} \\
&= \min_{0\leq\boldsymbol{\eta}\leq 1,\, M=\boldsymbol{\eta}\boldsymbol{\eta}^T} \max_{0\leq\boldsymbol{\alpha}\leq C} \boldsymbol{\eta}^T(\boldsymbol{\alpha} - \mathbf{e}) - \frac{1}{2}\boldsymbol{\alpha}^T(G \circ M)\boldsymbol{\alpha} \\
&\quad \text{with } M \succeq \boldsymbol{\eta}\boldsymbol{\eta}^T = M - \boldsymbol{\eta}\boldsymbol{\eta}^T \succeq 0 \\
&= \min_{0\leq\boldsymbol{\eta}\leq 1,\, M\succeq\boldsymbol{\eta}\boldsymbol{\eta}^T} \max_{0\leq\boldsymbol{\alpha}\leq C} \boldsymbol{\eta}^T(\boldsymbol{\alpha} - \mathbf{e}) - \frac{1}{2}\boldsymbol{\alpha}^T(G \circ M)\boldsymbol{\alpha}
\end{aligned}
\tag{3.6}
$$

This problem is a convex optimization problem, however the optimization variable M is not a vector but a positive semidefinite matrix, such kind of problems are solved by quadratic-programming solvers. Xu mentions that the proposed method is robust, but that it is not possible to identify outliers by evaluating the values of η_i after solving the optimization problem. Therefore Xu proposes to drop the term $1 - \eta$ in formula 3.5 such that η_i serves as a weight factor for each training instance $\mathbf{x}_i$ yielding the robust outlier detection algorithm. A similar approach has been applied by Zhou et al. [30] and will be explained later.

3.3 Truncated Hinge Loss Function

The truncated hinge loss function is an alternative way to define a robust loss function and is usually implemented by combining two hinge loss functions as shown in figure 3.1(c). It was first applied to the support vector machine by Collobert et al. [10] in the context of transductive SVMs in order to reduce the number of support vectors by removing errors from the training set. Wu et al. [26] extend the approach by Collobert to multiclass support vector machines. Wang et al. [25] on the other hand combine two Huber loss functions so that the edges of the truncated hinge loss function are smoothed. All three approaches use the constrained concave-convex procedure (CCCP) to solve the optimization problem as described in equation 3.8. Assuming that the objection function $J(\mathbf{w})$ can be split into a convex part $J_{vex}(\mathbf{w})$ and a concave part $J_{cave}(\mathbf{w})$, the CCCP minimizes the objective function $J(\mathbf{w})$ in an iterative way as shown in algorithm 1.

The optimization problem based on the truncated hinge loss function is then stated as a difference between two hinge loss functions H_1 and H_s.

$$
\min_{\mathbf{w}} \underbrace{\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i H_1\left(y_i f\left(\mathbf{x}_i\right)\right)}_{J_{vex}(\mathbf{w})} \underbrace{- C\sum_i H_s\left(y_i f\left(\mathbf{x}_i\right)\right)}_{J_{cav}(\mathbf{w})} \tag{3.8}
$$

Algorithm 1 CCCP algorithm

Initialize $\mathbf{w}^0$

repeat

$$\mathbf{w}^{t+1} = \min_{\mathbf{w}} J_{vex}(\mathbf{w}) + J'_{cave}(\mathbf{w}) \cdot \mathbf{w} \tag{3.7}$$

until convergence of $\mathbf{w}^t$

Collobert as well as Wu solve the optimization problem stated in equation 3.8 by applying CCCP as shown in algorithm 2.

Algorithm 2 CCCP for solving the ramp loss problem

$\boldsymbol{\beta}^0 = 0$

repeat

Solve the convex optimization problem

$$\begin{aligned} & \max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & -\beta_i^{t-1} \leq \alpha_i \leq C - \beta_i^{t-1} \\ & \sum_i y_i \alpha_i = 0 \end{aligned} \tag{3.9}$$

Compute b^t using the unbounded support vectors $0 < \alpha_i^t < C \Rightarrow y_i f(\mathbf{x}_i) = 1$

Update

$$\beta_i^t = \begin{cases} C & y_i f(\mathbf{x}_i) < s \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

until $\boldsymbol{\beta}^t = \boldsymbol{\beta}^{t-1}$

The advantage of solving this problem in the dual is, that standard SVM solvers can be used to solve the convex problem. Collobert proposes to train the initial classifier with $\beta_i = 0$ on a subset of the training data. This corresponds to the standard training procedure based on the hinge loss function. Based on this initial classifier all points which have a distance from the margin larger than s will be removed by setting $\beta_i = C$ and thus consequently setting $\alpha_i = 0$ in the next convex training step. Wu applies CCCP in the primal in order to solve the optimization problem with a Newton-type algorithm and sets $\beta_i = 0$ in equation 3.3 for those points who are regarded as outliers.

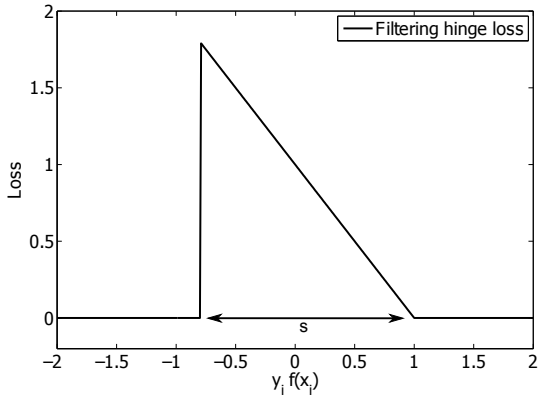


Figure 3.2: The filtering hinge loss function

3.4 Integrated Outlier Filtering

The integrated outlier filtering approach by Zhou [30] is an implementation of the robust outlier detection method proposed by Xu et al. [28] using the loss function shown in Figure 3.2. In order to show the equivalence we continue to use the notation introduced in section 3.2. The optimization problem is stated as

$$\begin{aligned} \min_{w} \min_{\eta_i \in \{0,1\}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_i \eta_i \xi_i \\ \text{subject to} \quad & \sum_i \eta_i \geq M \end{aligned} \tag{3.11}$$

where η_i is a binary filtering variable and M the number of non-outliers in the training set. This problem is then transformed into a semidefinite program by relaxing the binary constraint on η to $0 \leq \eta_i \leq 1$, as shown above

$$\begin{aligned} & \min_{0 \leq \boldsymbol{\eta} \leq 1} \min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \eta_i \left[1 - y_i \mathbf{x_i}^T \mathbf{w}\right]_+ \\ = & \min_{0 \leq \boldsymbol{\eta} \leq 1} \max_{0 \leq \boldsymbol{\alpha} \leq C} \boldsymbol{\eta}^T(\boldsymbol{\alpha}) - \frac{1}{2}\boldsymbol{\alpha}^T\left(\mathbf{Y}\mathbf{X}^T\mathbf{X}\mathbf{Y} \circ \boldsymbol{\eta}\boldsymbol{\eta}^T\right)\boldsymbol{\alpha} \\ & \text{subject to} \quad 0 \leq \boldsymbol{\alpha}_i \leq C,\ \mathbf{e}^T\boldsymbol{\eta} \geq M \\ & G = \mathbf{Y}\mathbf{X}^T\mathbf{X}\mathbf{Y} \\ = & \min_{0 \leq \boldsymbol{\eta} \leq 1,\ M \succeq \boldsymbol{\eta}\boldsymbol{\eta}^T} \max_{\boldsymbol{\alpha}} \boldsymbol{\eta}^T(\boldsymbol{\alpha} - \mathbf{e}) - \frac{1}{2}\boldsymbol{\alpha}^T(G \circ M)\boldsymbol{\alpha} \\ & \text{subject to} \quad 0 \leq \boldsymbol{\alpha}_i \leq C,\ \mathbf{e}^T\boldsymbol{\eta} \geq M \end{aligned} \tag{3.12}$$

In addition to this approach, which Zhou refers to as semi-definite programming robust SVM, Zhou proposes a multi-stage relaxation of the semi-definite programming based formulation based on CCCP as shown in Algorithm 3.

The Multi-stage robust SVM algorithm trains a support vector machine iteratively and removes outliers during each iteration. Since the derivative of the loss

Algorithm 3 Multi stage relaxation of integrated outlier filtering

$\boldsymbol{\eta}^0 = \mathbf{1}$

repeat

Solve the convex optimization problem

$$\begin{aligned} L_D^t &= \max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \eta_i \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C \end{aligned} \tag{3.13}$$

Update, sort $u_i = y_i f(\mathbf{x}_i)$ in ascending order

$$\eta_i^t = \begin{cases} 0 & \text{if the rank of } u_i > M \\ 1 & \text{otherwise} \end{cases} \tag{3.14}$$

until $L_D^{t-1} - L_D^t$ is sufficiently small

function of this algorithm is equal to the derivative of the truncated hinge loss discussed in section 3.3 both approaches are equal. This can also be seen by analyzing the algorithms. Setting $\eta_i = 0$ in algorithm 3 has the same effect as choosing $\beta_i = C$ in algorithm 2. Both approaches differ only in how they identify outliers. Zhou assumes that the number of non-outliers M is known beforehand and removes the outliers according to their rank, whereas Wu uses a threshold on the distance to the margin in order to identify outliers. Another difference is that the iterative removal of outliers according to their rank converges after M instances are left in the training set, the threshold criteria of the truncated hinge loss however only removes points according to the specified threshold.

4. EXPERIMENTS

The different methods to make support vector machine robust against noise by using modified loss functions were introduced in the last chapter. In this chapter the performance of the methods based on the truncated hinge loss function which solve the optimization problem in the dual is analyzed. The motivation to analyze this type of approaches is that they can be easily adopted to highly efficient SVM solvers as LibSVM. The approaches based on the truncated hinge loss function proposed by Collobert [10] and the "Integrated outlier filtering for large margin training" proposed by Zhou [30] fulfill this requirement. Collobert removes all training instances with a distance to the margin greater than a fixed value, Wu which implements the same approach in the primal proposes $s = -1$ as a threshold. Zhou on the other hand removes a certain percentage of the largest margin violators and states that the optimal amount of outliers to remove can be estimated by cross-validation.We will refer to the two approaches in the following as threshold filtering and rank filtering in our tables and plots. Both methods use the truncated hinge loss function in order to remove outliers.

The objective of our experiments is to determine if the classification accuracy of the robust methods is higher than the accuracy of a standard support vector machine in the presence of noise during training. Furthermore we want to examine if it is possible to determine the optimal robustness parameter for both methods by applying cross-validation as stated by Zhou [30]. The experimental study is designed according to the guidelines for machine learning experiments described by Alpaydin [1]. Following those recommendations we now give an overview over the central points of the experimental study.

We test the hypothesis that the robust algorithms perform better than the standard support vector machine in the presence of noise. The primary response variable used to test the hypothesis is the classification accuracy of the different algorithms on noisy data sets which we acquire in our experiments. Since the robust algorithms rely on implicit outlier detection and their removal, we also measure the accuracy of the outlier removal process using the F1-score as response variable. Furthermore the number of support vectors is selected as measure of the classifiers complexity.

Data set	Dimensionality	#Training	#Test
UCI Breast cancer	10	383	95
UCI Diabetes	8	429	108
UCI Ionosphere	34	202	50
UCI Liver disorders	6	236	58
UCI Fourclass	2	492	122
UCI Heart	13	192	48
UCI Sonar	60	156	38

Table 4.1: Dimensionality and size of the training and test sets used for evaluation

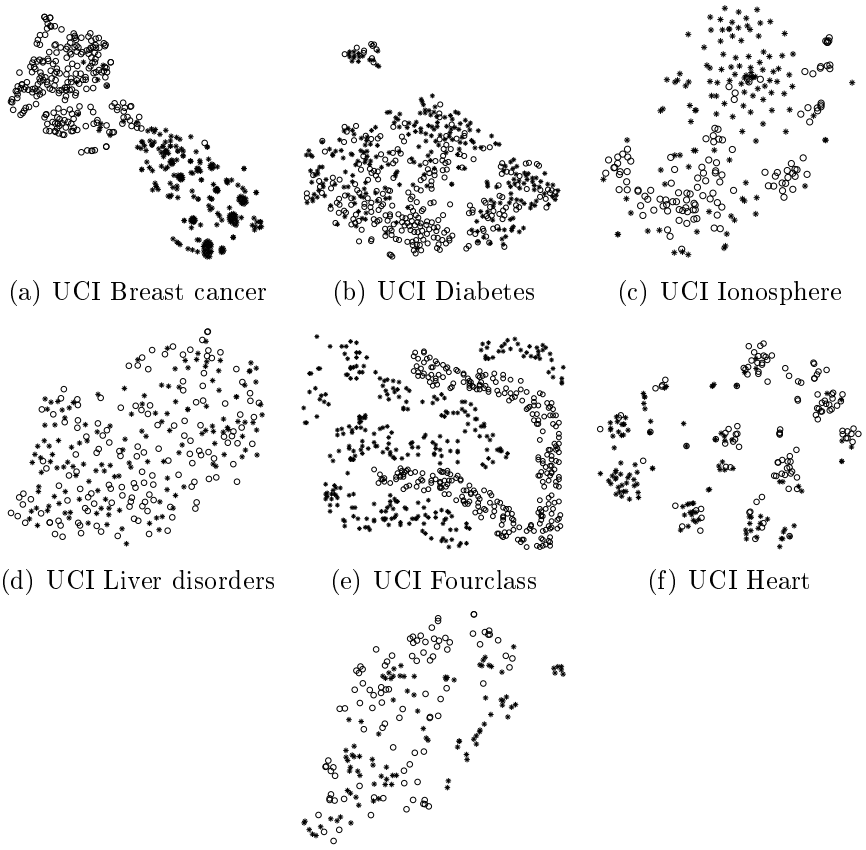
The support vector machines, and therefore the response variables, are generally influenced by a variety of factors. The controllable factors of our study are the cost parameter C of the soft-margin SVM, the parameter for the RBF kernel and the robustness parameter controlling the removal of outliers. The scale of the features and the ratio between the number of samples per class are also well known factors [3] [16] which affect the performance of support vector machines. In the following we describe the data sets used, how they are are preprocessed and resampled as well as the experimental strategy used to address the controllable factors.

4.1 Data sets, Resampling and Preparation

Seven different data sets representing binary classification problems are selected for the evaluation of the algorithms, their main characteristics are shown in table 4.1. The data sets are provided by the UCI machine learning database [13] and were retrieved from the collection of data sets provided at the LibSVM website [17]. The UCI data sets are widely used in publications and can be considered as standard evaluation data sets. The dimensionality as well as the size of the data sets selected is small to medium and all data sets are prone to noise. Figure 4.1 illustrates the distribution of the samples after applying t-Distributed Stochastic Neighbor Embedding as dimensionality reduction technique. As we can see from the figure, the two data sets UCI Breast cancer and UCI Fourclass are special since they are almost completely separable.

Regarding the data preparation it is necessary to scale each feature of the data sets such that they lie in the same interval as proposed by [16]. Since the data sets provided by the LibSVM website are standardized to a range of $[-1, +1]$ no further data preprocessing is applied.

The seven data sets selected are not divided into training and test sets, we therefore apply a resampling in order to split the data into training and test sets. Before dividing the data sets we stratify them such that both of the classes are represented by the same number of samples. The stratified data sets are then resampled by applying the five-fold cross-validation method as proposed by [18]. We set the replication number to 100 such that 500 training and test set pairs are generated for each UCI data set. Since we use five-fold cross-validation 80% of the original data



(a) UCI Breast cancer (b) UCI Diabetes (c) UCI Ionosphere

(d) UCI Liver disorders (e) UCI Fourclass (f) UCI Heart

(g) UCI Sonar

Figure 4.1: Visualization of the data sets used in the experiments

set size is used for training and 20% for testing. For each of the 100 cross-validation sets consisting of five training and test set pairs, the test sets are mutually exclusive. After collecting the response variables for each cross-validation set, we average their values yielding an overall number of 100 measurements.
In order to study how much the classification accuracy is affected by label noise, we generate 100 label noise realizations by randomly flipping 5% to 15% of the class labels for each data set as done in the experiments by Zhou. Additionally to random label noise we generate 100 instances of adversarial label noise. Different from randomly chosing the labels to flip we first train a SVM on the unpolluted dataset and flip the labels of the instances of class 1 which have the maximum positive distance from the decision boundary. By doing so the instances with the flipped labels can be regarded as outliers with respect to class 2. After generating the noise realizations we apply the same cross-validation method yielding 500 training and test set pairs for each noise level,data set and noise-type. We store all training and test sets generated in order to provide the different algorithms with the same training and test data following the blocking approach.

4.2 Experimental Strategy

By scaling and stratifying the data two influencing factors are removed. The remaining factors are the cost parameter C of the SVM, the parameter for the RBF kernel and the robustness parameter. In the following we describe how the best parameter combination is selected.
Determining the best cost-,kernel- and robustness parameters in terms of classifier accuracy is costly. A commonly used method to determine the optimal parameter combination is the grid search which simply tests all combinations of parameters in a given range. A more sophisticated approach is unconstrained nonlinear optimization. It takes the value of the objective function into account when selecting the next combination of parameters subject to test. We compared the results of the grid search and nonlinear optimization in terms of computational complexity as well as accuracy and decided to use grid search.

As shown above the grid search tests all parameter combinations in a given range by training the classifier on a training set and evaluating its performance on a validation set. For each run in our experiments we randomly select 20% of the training data as a tuning set and apply 5-fold cross-validation during the gridsearch. That is five different training and validation sets for the gridsearch are generated based on the tuning set. Once the optimal parameter configuration is found we train the final classifier on the whole training set and test its classification accuracy on the independent test.

Algorithm 5 Gridsearch algorithm to determine the optimal parameter setting for a RBF kernel.

```
for log2c = −5 → 7 do
    for log2gamma = −7 → 5 do
        do 5-fold cross-validation
        if average accuracy over 5-fold cv ≥ best accuracy found so far then
            update optimal parameter set
        end if
    end for
end for
```

In order to determine if the optimal robustness parameter can be selected by grid search, we extend the above algorithm by an outer loop searching a specified range. For rank based filtering we select the range as $s = 5 - 50\%$. This range is much larger than the amount of added noise, since we do not want to introduce information leakage. By analyzing the optimal values we will be able to judge if the optimal parameters found are related to the noise added. For threshold based filtering we choose a range of $[-1000, -2 : 0.5 : 0]$. Setting the treshold value to -1000 corresponds to not removing any training instances, training instances at $s = 0$ are situated directly at the decision boundary.

4.3 Implementation

Both robust support vector approaches are implemented in Matlab using the Matlab-Interface of LibSVM-Weights-3.12[7]. We choose the standard LibSVM 3.12 [7] implementation as the reference algorithm in our experiments. All tests are conducted on a Core i5 3.1 GHz computer with 8GB RAM, the statistical analysis is performed using the Statistical Toolbox of Matlab 2011b.

5. RESULTS

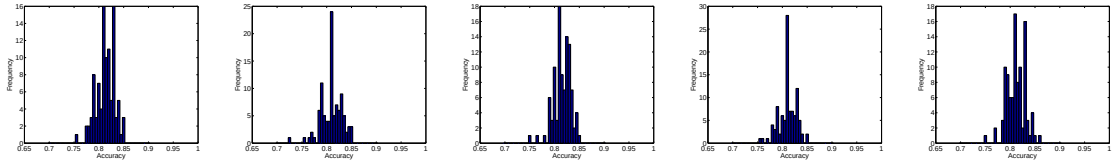


Figure 5.1: Histograms of the accuracy measures of the tested methods on the Ionosphere data set contamined by 10% label noise using a linear kernel.

After describing the experimental setup and strategy we will now discuss the results of the experiments. We first examine the distribution of classification accuracy measurements. Figure 5.1 shows the accuracy histograms of the different approaches tested. As we can see the accuracy values are situated in the same range, but their distributions differ.

In order to compare the measurement values, especially when their differences are small, we apply a statistical significance test to ensure that the differences between the algorithms are meaningful. We choose the Wilcoxon signed-rank test with $p < 0.05$ as an non-parametric statistic significance test. This paired difference test evaluates whether or not the difference between the two series of measurements originate from a distribution with zero median. Due to the fact that we use the same training and test sets for each algorithm, we obtain paired samples and can therefore apply this type of test. All values reported are the median values of 100 measures. In the tables showing the accuracy values of the different approaches we print those values in boldface with $p < 0.05$.

We now present the individual results. We first compare the results of the linear and the RBF kernel. Afterwards we compare the results for the robust methods with a fixed robustness parameter and a cross-validated parameter. All measurement tables are attached in the appendix.

As we can see in figure 5.2(a) the reference LibSVM is only mildly affected by label noise given a linear kernel and a nearly linearly separable dataset like UCI Breast cancer. Without any outliers LibSVM yields an accuracy of 96,45% and

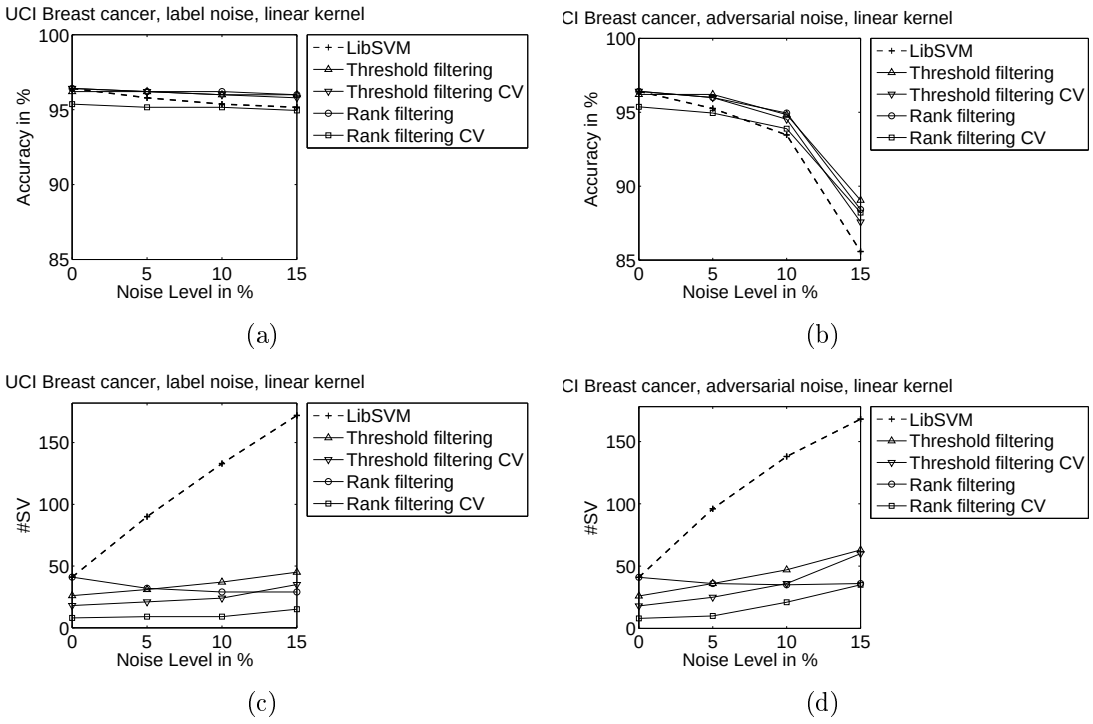


Figure 5.2: Classification accuracy and number of support vectors for the UCI Breast cancer dataset using a linear kernel.

by adding 15% of label noise the accuracy drops by 1,26%. In such a setting all outliers introduced by label noise are situated at the wrong side of of the decision surface and can be effectively filtered. Hence the accuracy is improved by the robust methods and the number of support vectors is reduced as shown in figure 5.2(c). Only the rank based filtering with cross-validated robustness parameter shows a accuracy worse than LibSVM. The F1-scores of the robust methods which are able to improve the accuracy, shown in table A.5, are high with a value of approximately 0,87. The higher the (F1-score $\in [0, 1]$) is, the better both precision and recall are. That means, most of the points filtered by the robust methods are in fact outliers introduced by the noise. The slope of the accuracy curves of the robust methods is also less steep than that of LibSVMs , which is reflected by the value of the absolute change in accuracy Δ_{Acc} in tables A.1 and A.3. In case of the adversarial noise the accuracy of LibSVM degrades more, especially for 15% of adversarial label noise. In this case the robust methods still improve the accuracy with respect to LibSVM but fail to detect the introduced noise correctly (F1-scores approximately 0,59).

Let us now analyze the performance of the classifiers on the UCI Liver disorders dataset, as shown in figure 5.3(a). It is interesting to observe that the accuracy of the classifier is improved if the robust methods are applied to the noiseless dataset. However the slopes of the accuracy curves are similar. Consequently the improve-

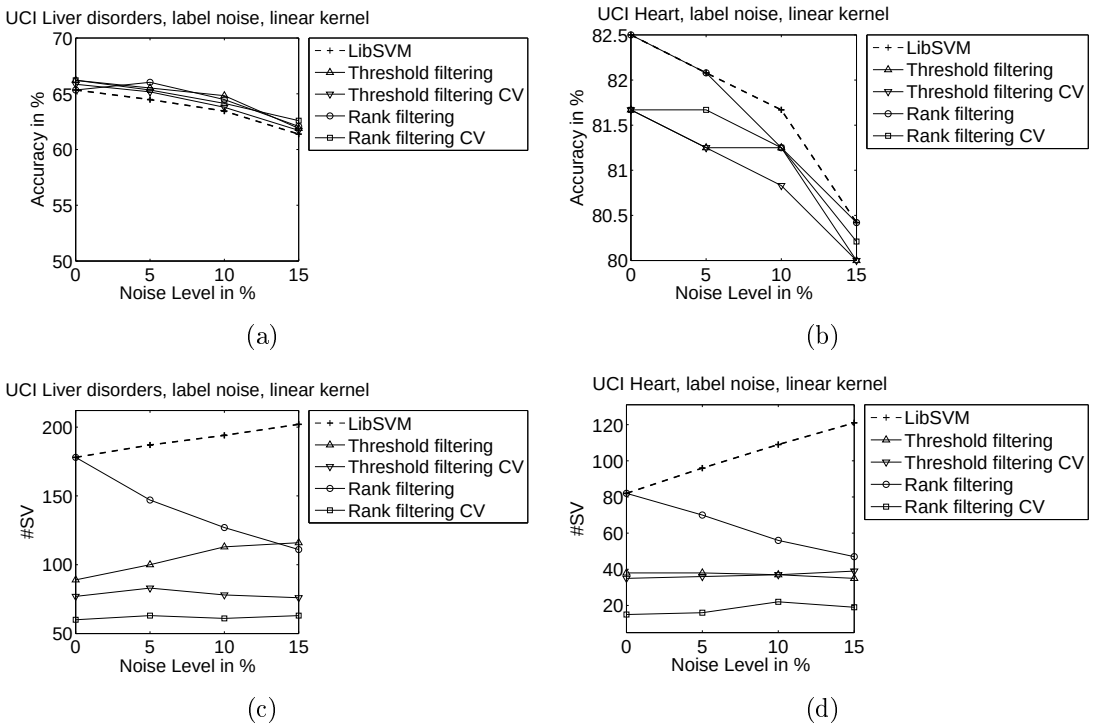


Figure 5.3: Classification accuracy and number of support vectors for the UCI Liver disorders and UCI Heart dataset using a linear kernel.

ment of accuracy is in this case linked to the removal of valid training samples, which is also reflected by low F1-scores. In figure 5.3(b) we can see an example where the application of the robust methods decreases the accuracy of the classifier. The F1-scores for UCI Heart are higher than those for UCI Liver disorders, but the removed points affect the classifiers performance much more than in the case of UCI Liver disorders. Therefore the improvement of accuracy is dependent of the dataset.

In order to summarize the results for the tests using the linear kernel we can conclude that from the seven datasets tested three of the four tested approaches showed an significant and consistent improvement in accuracy over the whole range of noise and a high F1-score on the UCI Breast cancer dataset. In all other cases the accuracy either declined or was mainly caused by the removal of valid training samples, yielding classifiers with a much smaller number of support vectors than LibSVM.

Different from the results using a linear kernel, the accuracy of the RBF kernel based robust approaches differs only slightly from the accuracy of LibSVM and the difference in number of support vectors is also smaller, since less training samples are removed. From the seven datasets used for testing, the Fourclass dataset contamined by label noise is the only case where the accuracy is significantly improved over the whole range of noise. This dataset is completely separable using the RBF kernel and

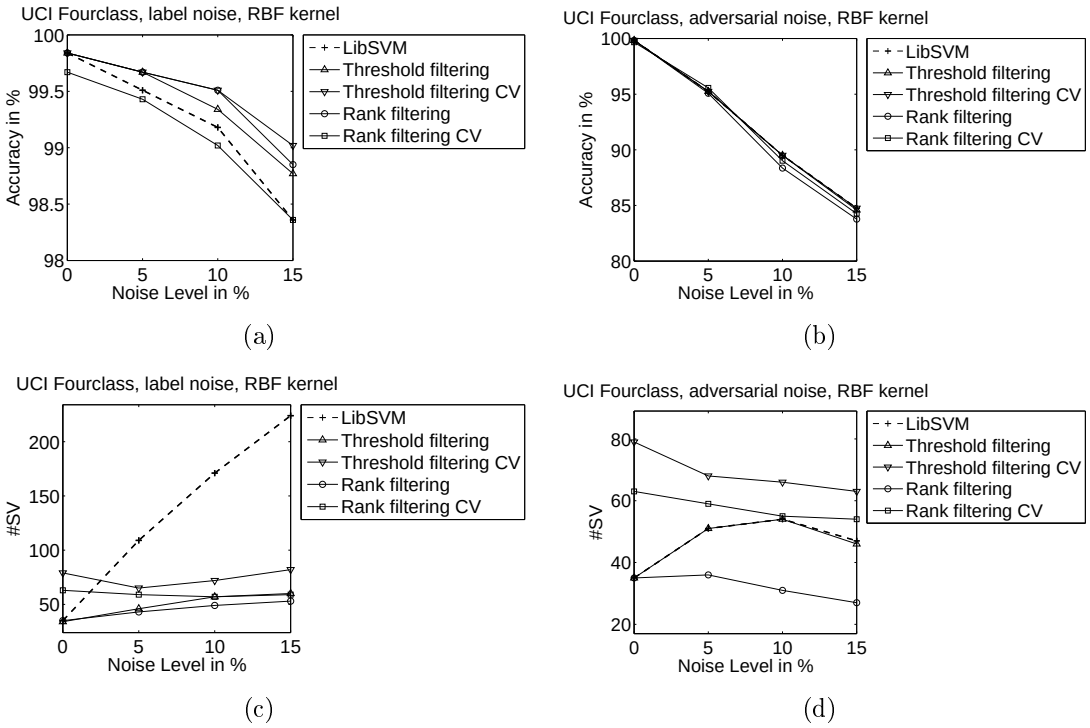


Figure 5.4: Classification accuracy and number of support vectors for the UCI Fourclass dataset using a RBF kernel.

the high F1-scores of approximately 0,96 indicate that the outliers were successfully identified and removed. For all other datasets the relative changes in accuracy are either not consistent, that is positive and negative over the range of noise, or the F1-score is very low such that non-noise training samples had been removed. We summarize that by using of an RBF kernel the number of removed training samples is reduced and the relative difference between LibSVM and the robust approaches is reduced.

Let us finally compare the results for the robust methods with and without cross-validated robustness parameter. The differences between the cross-validated robustness parameters and the standard parameters of threshold based filtering are random. That means on some datasets it is beneficial to choose $s = -1$ on other datasets a cross-validated value of s performs slightly better. In case of the rank based filtering trying to estimate the robustness parameter by cross-validation fails and yields worse results than setting the percentage to the amount of noise introduced. We can conclude that estimating the robustness parameter by cross-validation, either the threshold value or the percentage of points to remove, does not increase the accuracy for the robust approaches. This contradicts the statement of Zhou et al. [30] that the robustness parameter can be determined by cross-validation.

5.1 Discussion

The experimental results have shown that an improvement of robustness by removing the introduced label and adversarial noise is only feasible if the dataset is separable by the kernel used in the robust SVM. For some non-separable datasets the resulting sparse classifier yields a better performance as for example demonstrated at the UCI Liver disorders dataset. In other tests, as for example UCI Heart, the performance of the classifier degrades by applying the robust methods. The reason for this effect is that the robust methods rely in their initial filtering step of CCCP on a standard SVM in order to decide which points are outliers. Based on this estimate of the decision boundary outliers are iteratively removed and the decision boundary is adapted. If the standard SVM used for the first step yields a decision boundary which is able to detect the outliers correctly, they can be removed and accuracy is increased. This is the case for separable datasets.

If the initial decision boundary is influenced by the outliers in the training set in such a way that also valid training samples are identified as outliers, it depends on the composition of the datasets if the accuracy can be improved or not. Due to this dependency on the datasets we conclude that the methods tested are not robust in the sense that they reduce the sensitivity of a standard SVM on outliers in general. Their effect is rather a reduction of support vectors in order to make the SVM more sparse and an improvement in accuracy is not guaranteed for an arbitrary dataset.

Let us now explain why it is not possible to tune the robustness parameter by using cross-validation. During the grid search we are determining the optimal robustness parameter in order to get the best accuracy with respect given tuning set. Since the tuning set is contamined by outliers, cross-validation returns a robustness parameter which yields the best classifier under the assumption that the tuning contamined by outliers represents the true data distribution. However our aim is to remove outliers and to improve the classifier with respect to the original dataset, that is why the optimal robustness parameters can not be reliably estimated in the presence of outliers. Nonetheless we were able to observe that threshold based outlier removal performs better than the rank based method. We conclude that the process of estimating the threshold value is more stable than estimating the percentage of training samples to remove.

6. CONCLUSION

By summarizing our findings from chapter 5 we can conclude that the application of the robust methods based on implicit outlier filtering by applying a truncated loss function is risky. The advantage of the robust methods is that they yield sparse classifiers, which reduces the time needed to classify an instance. But sparseness of the classifier does not guarantee robustness against outliers. If the dataset is nearly separable the robust methods tested perform as expected and reliably remove the outliers. However if the dataset is not separable, the application of the robust methods based on the truncated hinge loss function may degrade the classifiers accuracy. Due to this we recommend to use the standard support vector machine and apply filtering of the outliers as a preprocessing step.

BIBLIOGRAPHY

[1] Alpaydin, E. (2010). *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. MIT Press.

[2] Bartlett, P. L., & Maass, W. (2003). Vapnik-Chervonenkis dimension of neural nets. In *The Handbook of Brain Theory and Neural Networks*. MIT Press.

[3] Ben-Hur, A., & Weston, J. (2010). A User's Guide to Support Vector Machines Data Mining Techniques for the Life Sciences. Humana Press.

[4] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[5] Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[6] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*.

[7] Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*.

[8] Chapelle, O., Schölkopf, B., & Zien, A. (Eds.) (2006). *Semi-Supervised Learning*. MIT Press.

[9] Christmann, A., & Steinwart, I. (2004). On robustness properties of convex risk minimization methods for pattern recognition. *Journal of Machine Learning Research*.

[10] Collobert, R., Sinz, F., Weston, J., & Bottou, L. (2006). Trading convexity for scalability. In *Proceedings of the 23rd international conference on Machine learning*.

[11] Cortes, C., & Vapnik, V. (1995). Support-vector networks. In *Machine Learning*, (pp. 273–297).

[12] Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

[13] Frank, A., & Asuncion, A. (2010). Uci machine learning repository. URL `http://archive.ics.uci.edu/ml`

[14] Hable, R., & Christmann, A. (2011). On qualitative robustness of support vector machines. *Journal of Multivariate Analysis*.

[15] Hastie, T., Tibshirani, R., & Friedman, J. H. (2003). *The Elements of Statistical Learning*. Springer.

[16] Hsu, C.-w., Chang, C.-c., & Lin, C.-j. (2010). A practical guide to support vector classification. *Bioinformatics*.

[17] Lin, C.-J. (2012). Libsvm data: Classification (binary class).
URL `http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html`

[18] Salzberg, S. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*.

[19] Schölkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*.

[20] Schölkopf, B., & Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

[21] Steinwart, I., & Christmann, A. (2008). *Support Vector Machines*. Springer.

[22] Vapnik, V. (1998). *Statistical learning theory*. Wiley.

[23] Vapnik, V. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks*.

[24] von Luxburg, U., & Schölkopf, B. (2011). Statistical learning theory: Models, concepts, and results. In *Handbook for the History of Logic, vol. 10*. Elsevier.

[25] Wang, L., Jia, H., & Li, J. (2008). Training robust support vector machine with smooth ramp loss in the primal space. *Neurocomputing*, *71*(13-15), 3020 – 3025.

[26] Wu, Y., & Liu, Y. (2007). Robust truncated-hinge-loss support vector machines. *Journal of the Acoustical Society of America*.

[27] Xu, H., Caramanis, C., & Mannor, S. (2009). Robustness and regularization of support vector machines. *Journal of Machine Learning Research*.

[28] Xu, L., Crammer, K., & Schuurmans, D. (2006). Robust support vector machine training via convex outlier ablation. In *Proceedings of the 21st national conference on Artificial intelligence*.

[29] Yang, X., Song, Q., & Wang, Y. (2007). A weighted support vector machine for data classification. *International Journal of Pattern Recognition and Artificial Intelligence*.

[30] Zhou, X.-c., Shen, H.-b., & Ye, J.-p. (2011). Integrating outlier filtering in large margin training. *Journal of Zhejiang University*.

A. APPENDIX

Noise Level	LibSVM	Threshold filtering	Threshold filtering CV	Rank filtering	Rank filtering CV
UCI Breastcancer					
0%	96,42%	**-0,21%**	0,00%	0,00%	**-1,05%**
5%	95,79%	**0,42%**	**0,42%**	**0,42%**	**-0,63%**
10%	95,37%	**0,63%**	**0,63%**	**0,84%**	**-0,21%**
15%	95,16%	**0,84%**	**0,63%**	**0,84%**	-0,21%
Δ_{Acc}	1,26%	0,21%	0,63%	0,42%	0,42%
UCI Diabetes					
0%	72,52%	**-0,74%**	**-0,56%**	0,00%	-0,18%
5%	72,15%	**-0,19%**	**0,00%**	**-0,19%**	0,19%
10%	72,15%	**-0,75%**	**-0,56%**	**-0,19%**	**-0,19%**
15%	72,15%	**-0,56%**	**-0,37%**	**-0,19%**	**-0,37%**
Δ_{Acc}	0,37%	0,19%	0,18%	0,56%	0,56%
UCI Ionosphere					
0%	82,80%	**-0,40%**	-0,20%	0,00%	**-0,40%**
5%	82,00%	**0,00%**	0,40%	0,20%	**-0,20%**
10%	81,60%	**-0,40%**	0,00%	**-0,40%**	-0,40%
15%	79,80%	0,20%	**0,60%**	0,20%	**1,00%**
Δ_{Acc}	3,00%	2,40%	2,20%	2,80%	1,60%
UCI Liver disorders					
0%	65,34%	**0,87%**	**0,52%**	0,00%	**0,87%**
5%	64,48%	**1,04%**	**0,69%**	**1,55%**	**0,86%**
10%	63,45%	**1,38%**	0,34%	**1,03%**	**0,69%**
15%	61,38%	**0,52%**	0,34%	**0,69%**	**1,21%**
Δ_{Acc}	3,96%	4,31%	4,14%	3,27%	3,62%
UCI Fourclass					
0%	71,31%	**5,25%**	**3,20%**	0,00%	**1,48%**
5%	71,64%	**4,75%**	**2,70%**	**2,13%**	**1,47%**
10%	71,64%	**4,67%**	**2,62%**	**3,11%**	**1,15%**
15%	71,80%	**3,94%**	**2,13%**	**2,63%**	**1,15%**
Δ_{Acc}	-0,49%	0,82%	0,58%	-3,12%	-0,16%
UCI Heart					
0%	82,50%	**-0,83%**	**-0,83%**	0,00%	**-0,83%**
5%	82,08%	**-0,83%**	**-0,83%**	0,00%	**-0,41%**
10%	81,67%	**-0,42%**	**-0,84%**	**-0,42%**	**-0,42%**
15%	80,42%	**-0,42%**	**-0,42%**	**0,00%**	**-0,21%**
Δ_{Acc}	2,08%	1,67%	1,67%	2,08%	1,46%
UCI Sonar					
0%	73,68%	**0,53%**	-0,52%	0,00%	-0,52%
5%	72,63%	**0,26%**	0,26%	0,00%	0,00%
10%	71,05%	**0,00%**	0,00%	-0,26%	-0,26%
15%	68,95%	**1,58%**	0,52%	**1,05%**	0,52%
Δ_{Acc}	4,73%	3,68%	3,69%	3,68%	3,69%

Table A.1: Classification accuracies using the linear kernel on datasets contamined by class noise

Noise Level	LibSVM	Threshold filtering	Threshold filtering CV	Rank filtering	Rank filtering CV
UCI Breastcancer					
0%	95,58%	0,00%	0,00%	0,00%	0,00%
5%	95,58%	-0,21%	0,21%	0,00%	-0,21%
10%	95,37%	-0,32%	0,00%	**-0,42%**	-0,21%
15%	94,95%	0,00%	**0,21%**	**-0,63%**	-0,21%
Δ_{Acc}	0,63%	0,63%	0,42%	1,26%	0,84%
UCI Diabetes					
0%	71,78%	**-0,38%**	**-0,38%**	0,00%	-0,19%
5%	71,40%	**-0,19%**	0,00%	**-0,19%**	-0,37%
10%	70,84%	-0,37%	**-0,37%**	**-0,47%**	**-0,56%**
15%	70,28%	0,19%	0,37%	-0,37%	-0,37%
Δ_{Acc}	1,50%	0,93%	0,75%	1,87%	1,68%
UCI Ionosphere					
0%	91,80%	-0,20%	**-0,20%**	0,00%	**-1,40%**
5%	89,60%	-0,20%	0,00%	**0,40%**	-0,40%
10%	87,60%	0,00%	0,00%	0,00%	**0,80%**
15%	86,00%	**0,00%**	0,40%	**1,00%**	0,60%
Δ_{Acc}	5,80%	5,60%	5,20%	4,80%	3,80%
UCI Liver disorders					
0%	66,38%	0,17%	0,17%	0,00%	**-0,52%**
5%	65,86%	-0,34%	-1,38%	0,00%	**-0,86%**
10%	64,14%	**0,69%**	0,00%	**-0,17%**	-0,35%
15%	62,41%	**0,35%**	0,35%	0,00%	0,00%
Δ_{Acc}	3,97%	3,79%	3,79%	3,97%	3,45%
UCI Fourclass					
0%	99,84%	0,00%	**0,00%**	0,00%	**-0,17%**
5%	99,51%	**0,16%**	0,16%	**0,16%**	**-0,08%**
10%	99,18%	**0,16%**	**0,33%**	**0,33%**	**-0,16%**
15%	98,36%	**0,41%**	**0,66%**	**0,49%**	0,00%
Δ_{Acc}	1,48%	1,07%	0,82%	0,99%	1,31%
UCI Heart					
0%	81,25%	**0,00%**	**-0,83%**	0,00%	**-0,42%**
5%	80,63%	-0,21%	**-0,63%**	-0,21%	-0,21%
10%	79,17%	0,00%	**0,41%**	0,00%	**0,41%**
15%	77,08%	**0,63%**	**0,84%**	0,00%	**1,67%**
Δ_{Acc}	4,17%	3,54%	2,50%	4,17%	2,08%
UCI Sonar					
0%	83,16%	0,00%	**-0,53%**	0,00%	**-4,21%**
5%	81,05%	-0,26%	**-1,58%**	-1,05%	**-4,73%**
10%	76,58%	-0,26%	**-1,32%**	-0,26%	**-3,16%**
15%	73,16%	0,00%	**-0,53%**	0,00%	**-2,63%**
Δ_{Acc}	10,00%	10,00%	10,00%	10,00%	8,42%

Table A.2: Classification accuracies using the RBF kernel on datasets contamined by class noise

Noise Level	LibSVM	Threshold filtering	Threshold filtering CV	Rank filtering	Rank filtering CV
UCI Breastcancer					
0%	96,42%	**-0,21%**	0,00%	0,00%	**-1,05%**
5%	95,26%	**0,95%**	**0,74%**	**0,74%**	**-0,31%**
10%	93,47%	**1,37%**	**1,06%**	**1,48%**	**0,42%**
15%	85,58%	**3,47%**	**2,00%**	**2,84%**	**2,63%**
Δ_{Acc}	10,84%	7,16%	8,84%	8,00%	7,16%
UCI Diabetes					
0%	72,52%	**-0,74%**	**-0,56%**	0,00%	-0,18%
5%	72,34%	**-0,56%**	**-0,28%**	**-0,38%**	0,00%
10%	69,91%	**-1,50%**	**-0,94%**	**-1,12%**	**-0,56%**
15%	56,82%	**3,55%**	**2,34%**	**2,62%**	**2,34%**
Δ_{Acc}	15,70%	11,41%	12,80%	13,08%	13,18%
UCI Ionosphere					
0%	82,80%	**-0,40%**	-0,20%	0,00%	**-0,40%**
5%	82,80%	**-0,80%**	**-0,40%**	**-0,40%**	**-0,40%**
10%	81,40%	**-0,20%**	**-0,20%**	**-1,00%**	**-0,40%**
15%	72,60%	**-0,60%**	0,20%	0,60%	1,40%
Δ_{Acc}	10,20%	10,40%	9,80%	9,60%	8,40%
UCI Liver disorders					
0%	65,34%	**0,87%**	**0,52%**	0,00%	**0,87%**
5%	62,59%	**-0,18%**	-0,52%	0,17%	-0,18%
10%	55,86%	**1,90%**	**1,04%**	**1,73%**	**0,35%**
15%	53,28%	**0,17%**	0,51%	**0,17%**	-0,18%
Δ_{Acc}	12,06%	12,76%	12,07%	11,89%	13,11%
UCI Fourclass					
0%	71,31%	**5,25%**	**3,20%**	0,00%	**1,48%**
5%	71,64%	**4,75%**	**2,54%**	**-0,16%**	**0,98%**
10%	55,25%	**5,08%**	**3,11%**	**5,65%**	**4,09%**
15%	50,00%	0,00%	0,00%	0,00%	0,00%
Δ_{Acc}	21,31%	26,56%	24,51%	21,31%	22,79%
UCI Heart					
0%	82,50%	**-0,83%**	**-0,83%**	0,00%	**-0,83%**
5%	81,04%	**-1,04%**	**-0,62%**	**-1,04%**	**-0,62%**
10%	77,08%	**0,42%**	0,42%	-0,20%	**0,84%**
15%	72,92%	0,41%	**0,41%**	0,41%	**1,25%**
Δ_{Acc}	9,58%	8,34%	8,34%	9,17%	7,50%
UCI Sonar					
0%	73,68%	**0,53%**	-0,52%	0,00%	-0,52%
5%	71,58%	**0,26%**	-0,53%	**0,53%**	0,00%
10%	69,47%	0,00%	**-1,05%**	**-1,05%**	0,53%
15%	61,05%	0,79%	**-1,05%**	**-1,05%**	1,06%
Δ_{Acc}	12,63%	12,37%	13,16%	13,68%	11,05%

Table A.3: Classification accuracies using the linear kernel on datasets contamined by adversarial noise

Noise Level	LibSVM	Threshold filtering	Threshold filtering CV	Rank filtering	Rank filtering CV
UCI Breastcancer					
0%	95,58%	0,00%	0,00%	0,00%	0,00%
5%	95,37%	0,21%	0,21%	-0,11%	0,00%
10%	94,74%	**-0,42%**	**0,21%**	-0,21%	-0,11%
15%	88,21%	0,00%	**1,26%**	0,11%	**1,16%**
Δ_{Acc}	7,37%	7,37%	6,11%	7,26%	6,21%
UCI Diabetes					
0%	71,78%	**-0,38%**	**-0,38%**	0,00%	-0,19%
5%	70,65%	**-0,56%**	0,00%	**-0,74%**	0,00%
10%	66,82%	**-0,28%**	**-0,46%**	**-0,65%**	-0,28%
15%	60,93%	-0,18%	-0,18%	-0,28%	**0,19%**
Δ_{Acc}	10,85%	10,65%	10,65%	11,13%	10,47%
UCI Ionosphere					
0%	91,80%	-0,20%	**-0,20%**	0,00%	**-1,40%**
5%	89,20%	**-0,20%**	0,00%	**0,40%**	-0,60%
10%	86,00%	0,00%	**0,60%**	0,40%	0,40%
15%	79,20%	0,00%	**1,20%**	**2,00%**	**2,80%**
Δ_{Acc}	12,60%	12,40%	11,20%	10,60%	8,40%
UCI Liver disorders					
0%	66,38%	0,17%	0,17%	0,00%	**-0,52%**
5%	63,45%	0,34%	-0,35%	0,34%	0,00%
10%	59,66%	0,17%	**-0,69%**	**-0,35%**	**-0,35%**
15%	56,55%	0,35%	**-0,69%**	0,00%	**-0,69%**
Δ_{Acc}	9,83%	9,65%	10,69%	9,83%	10,00%
UCI Fourclass					
0%	99,84%	0,00%	**0,00%**	0,00%	**-0,17%**
5%	95,25%	0,00%	0,00%	-0,17%	**0,32%**
10%	89,51%	0,00%	0,00%	**-1,15%**	**-0,49%**
15%	84,75%	-0,16%	**0,00%**	**-0,98%**	**-0,49%**
Δ_{Acc}	15,09%	15,25%	15,09%	16,07%	15,41%
UCI Heart					
0%	81,25%	**0,00%**	**-0,83%**	0,00%	**-0,42%**
5%	78,75%	0,42%	**0,42%**	-0,42%	**0,83%**
10%	75,42%	**0,41%**	**0,83%**	0,00%	**2,08%**
15%	70,83%	0,42%	**0,84%**	1,25%	**2,50%**
Δ_{Acc}	10,42%	10,00%	8,75%	9,17%	7,50%
UCI Sonar					
0%	83,16%	0,00%	**-0,53%**	0,00%	**-4,21%**
5%	77,63%	0,26%	-0,52%	0,26%	**-1,84%**
10%	70,26%	0,00%	**0,27%**	0,27%	0,79%
15%	61,84%	-0,26%	-1,05%	**0,27%**	**1,84%**
Δ_{Acc}	21,32%	21,58%	21,84%	21,05%	15,27%

Table A.4: Classification accuracies using the RBF kernel on datasets contamined by adversarial noise

Noise Level	LibSVM	Threshold filtering		Threshold filtering CV		Rank filtering		Rank filtering CV	
	#SV	# SV	F1	# SV	F1	# SV	F1	# SV	F1
UCI Breastcancer									
0%	41	26	0,00	18	0,00	41	0,00	8	0,00
5%	90	31	0,82	21	0,72	32	0,82	9	0,27
10%	133	37	0,88	24	0,82	29	0,87	9	0,43
15%	172	45	0,88	35	0,84	29	0,90	15	0,55
UCI Diabetes									
0%	264	146	0,00	121	0,00	264	0,00	55	0,00
5%	284	153	0,27	126	0,22	222	0,33	62	0,20
10%	305	146	0,39	125	0,32	191	0,42	82	0,32
15%	324	146	0,45	124	0,38	158	0,48	99	0,42
UCI Ionosphere									
0%	75	66	0,00	50	0,00	75	0,00	28	0,00
5%	90	66	0,44	50	0,37	61	0,47	28	0,21
10%	107	68	0,51	50	0,49	52	0,55	31	0,34
15%	121	72	0,53	52	0,57	47	0,60	33	0,44
UCI Liver disorders									
0%	178	89	0,00	77	0,00	178	0,00	60	0,00
5%	187	100	0,17	83	0,15	147	0,15	63	0,15
10%	194	113	0,24	78	0,23	127	0,24	61	0,24
15%	202	116	0,29	76	0,28	111	0,31	63	0,30
UCI Fourclass									
0%	297	64	0,00	71	0,00	297	0,00	59	0,00
5%	322	66	0,24	79	0,22	249	0,35	73	0,21
10%	341	57	0,38	70	0,35	207	0,44	77	0,34
15%	365	62	0,48	90	0,43	173	0,49	87	0,43
UCI Heart									
0%	82	38	0,00	35	0,00	82	0,00	15	0,00
5%	96	38	0,38	36	0,35	70	0,41	16	0,21
10%	109	37	0,52	37	0,48	56	0,54	22	0,36
15%	121	35	0,58	39	0,55	47	0,59	19	0,46
UCI Sonar									
0%	77	63	0,00	58	0,00	77	0,00	36	0,00
5%	90	67	0,27	60	0,25	71	0,30	38	0,16
10%	100	67	0,36	60	0,35	67	0,37	40	0,29
15%	107	68	0,38	59	0,38	60	0,40	41	0,35

Table A.5: Number of support vectors and F1-scores using the linear kernel on datasets contamined by class noise

Noise Level	LibSVM	Threshold filtering		Threshold filtering CV		Rank filtering		Rank filtering CV	
	# SV	# SV	F1	# SV	F1	# SV	F1	# SV	F1
UCI Breastcancer									
0%	85	79	0,00	66	0,00	85	0,00	25	0,00
5%	131	93	0,66	69	0,68	76	0,70	26	0,15
10%	169	107	0,67	77	0,76	81	0,76	35	0,32
15%	206	132	0,67	86	0,78	85	0,79	49	0,42
UCI Diabetes									
0%	266	208	0,00	147	0,00	266	0,00	71	0,00
5%	288	222	0,26	157	0,21	235	0,28	76	0,17
10%	307	227	0,34	162	0,31	205	0,37	94	0,28
15%	324	232	0,37	171	0,38	179	0,42	103	0,36
UCI Ionosphere									
0%	107	106	0,00	101	0,00	107	0,00	66	0,00
5%	118	114	0,13	107	0,28	103	0,34	62	0,18
10%	135	127	0,18	113	0,37	102	0,42	71	0,29
15%	145	137	0,18	118	0,40	103	0,44	70	0,37
UCI Liver disorders									
0%	163	129	0,00	104	0,00	163	0,00	79	0,00
5%	172	138	0,16	104	0,16	142	0,17	81	0,15
10%	179	147	0,20	109	0,24	125	0,26	85	0,24
15%	186	153	0,20	111	0,27	114	0,31	88	0,30
UCI Fourclass									
0%	35	34	0,00	79	0,00	35	0,00	63	0,00
5%	109	46	0,98	65	0,95	43	0,96	59	0,36
10%	171	57	0,98	72	0,94	49	0,97	57	0,56
15%	224	60	0,96	82	0,92	53	0,96	59	0,61
UCI Heart									
0%	90	61	0,00	52	0,00	90	0,00	23	0,00
5%	105	64	0,31	55	0,31	81	0,34	24	0,18
10%	120	65	0,44	59	0,44	73	0,45	28	0,31
15%	130	71	0,48	61	0,51	72	0,50	33	0,40
UCI Sonar									
0%	101	100	0,00	96	0,00	101	0,00	56	0,00
5%	109	104	0,16	98	0,18	98	0,26	59	0,16
10%	122	110	0,17	102	0,23	94	0,33	63	0,27
15%	126	115	0,18	106	0,24	91	0,35	65	0,33

Table A.6: Number of support vectors and F1-scores using the RBF kernel on datasets contamined by class noise.

Noise Level	LibSVM	Threshold filtering		Threshold filtering CV		Rank filtering		Rank filtering CV	
	# SV	# SV	F1	# SV	F1	# SV	F1	# SV	F1
UCI Breastcancer									
0%	41	26	0,00	18	0,00	41	0,00	8	0,00
5%	96	36	0,86	25	0,71	36	0,84	10	0,29
10%	138	47	0,86	36	0,75	35	0,82	21	0,42
15%	168	63	0,63	60	0,51	36	0,63	35	0,44
UCI Diabetes									
0%	264	146	0,00	121	0,00	264	0,00	55	0,00
5%	284	146	0,25	135	0,19	224	0,28	58	0,17
10%	298	151	0,23	129	0,21	185	0,24	70	0,24
15%	299	155	0,08	96	0,10	138	0,07	56	0,18
UCI Ionosphere									
0%	75	66	0,00	50	0,00	75	0,00	28	0,00
5%	90	71	0,43	49	0,38	65	0,46	29	0,23
10%	104	81	0,36	53	0,46	59	0,48	30	0,35
15%	115	93	0,28	46	0,40	52	0,43	26	0,41
UCI Liver disorders									
0%	178	89	0,00	77	0,00	178	0,00	60	0,00
5%	186	95	0,10	72	0,08	150	0,05	59	0,09
10%	172	91	0,03	65	0,02	110	0,00	49	0,04
15%	157	80	0,00	47	0,00	70	0,00	36	0,04
UCI Fourclass									
0%	297	86	0,00	71	0,00	297	0,00	59	0,00
5%	351	80	0,36	64	0,29	292	0,88	82	0,28
10%	395	190	0,18	47	0,17	250	0,21	70	0,23
15%	357	65	0,00	5	0,00	64	0,00	1	0,13
UCI Heart									
0%	82	52	0,00	35	0,00	82	0,00	15	0,00
5%	85	53	0,03	39	0,13	59	0,02	20	0,14
10%	86	53	0,15	39	0,20	37	0,19	17	0,21
15%	86	51	0,14	39	0,20	24	0,21	18	0,24
UCI Sonar									
0%	77	71	0,00	58	0,00	77	0,00	36	0,00
5%	94	81	0,32	61	0,32	78	0,36	40	0,21
10%	102	89	0,16	56	0,30	71	0,30	38	0,27
15%	103	92	0,03	41	0,13	59	0,15	32	0,26

Table A.7: Number of support vectors and F1-scores using the linear kernel on datasets contamined by adversarial noise.

Noise Level	LibSVM	Threshold filtering		Threshold filtering CV		Rank filtering		Rank filtering CV	
	# SV	# SV	F1	# SV	F1	# SV	F1	# SV	F1
UCI Breastcancer									
0%	85	79	0,00	66	0,00	85	0,00	25	0,00
5%	126	101	0,40	71	0,58	82	0,59	26	0,16
10%	159	127	0,37	70	0,66	77	0,69	34	0,33
15%	170	150	0,16	102	0,42	71	0,54	45	0,42
UCI Diabetes									
0%	266	208	0,00	147	0,00	266	0,00	71	0,00
5%	284	237	0,12	165	0,13	232	0,14	87	0,13
10%	289	250	0,04	198	0,08	201	0,10	110	0,15
15%	287	246	0,00	208	0,02	170	0,07	119	0,12
UCI Ionosphere									
0%	107	106	0,00	101	0,00	107	0,00	66	0,00
5%	119	113	0,20	103	0,37	102	0,41	68	0,18
10%	133	128	0,15	109	0,39	105	0,42	69	0,30
15%	146	141	0,12	121	0,37	105	0,42	66	0,36
UCI Liver disorders									
0%	163	129	0,00	104	0,00	163	0,00	79	0,00
5%	167	133	0,04	110	0,04	139	0,02	85	0,05
10%	156	130	0,00	107	0,01	108	0,02	74	0,06
15%	147	121	0,00	96	0,00	80	0,02	63	0,08
UCI Fourclass									
0%	35	35	0,00	79	0,00	35	0,00	63	0,00
5%	51	51	0,00	68	0,07	36	0,34	59	0,16
10%	54	54	0,00	66	0,01	31	0,26	55	0,14
15%	47	46	0,00	63	0,01	27	0,20	54	0,15
UCI Heart									
0%	90	74	0,00	52	0,00	90	0,00	23	0,00
5%	92	79	0,00	56	0,09	72	0,02	26	0,14
10%	99	84	0,08	62	0,15	58	0,16	27	0,21
15%	104	91	0,03	68	0,12	56	0,19	31	0,24
UCI Sonar									
0%	101	101	0,00	96	0,00	101	0,00	56	0,00
5%	116	113	0,07	103	0,26	104	0,37	62	0,19
10%	127	126	0,02	114	0,21	105	0,30	65	0,27
15%	134	134	0,00	119	0,06	102	0,22	62	0,28

Table A.8: Number of support vectors and F1-scores using the RBF kernel on datasets contamined by adversarial noise.