The Maximum-Margin Approach to Learning Text Classifiers
Methods, Theory, and Algorithms

Dissertation zur Erlangung eines

Doktors der Naturwissenschaften

der

Universität Dortmund
am Fachbereich Informatik

von

Dipl. Inform. Thorsten Joachims

Dortmund
August 2000

PRE-DEFENSE DRAFT

Tag der mündlichen Prüfung:

Dekan: Prof. Bernd Reusch

Gutachter: Prof. Katharina Morik, Prof. Tom Mitchell, Prof. Norbert Fuhr

Acknowledgments

Without each of the following people, this dissertation would have turned out differently. Or maybe, it would never have been started nor finished.

First, I would like to thank the two people who had the largest immediate influence on this work — Prof. Katharina Morik and Prof. Tom Mitchell. Prof. Morik was the person who got me excited about machine learning. It was her enthusiasm that made and still makes me want to understand the nature of learning. In particular, I thank her for teaching me about good and important research and for the guidance that shaped this dissertation without constraining it.

Prof. Mitchell was the person who introduced me to text learning during my time at Carnegie Mellon University. So the particular topic of this dissertation is due to him. I am very grateful for his advice and for the motivation he managed to get across despite the distance. I also thank Prof. Fuhr for his comments and his suggestions about this work.

Particular thanks go to Dr. Phoebe Sengers. She kept me happy and appropriately focused throughout this dissertation. Despite her adversion towards Greek letters, she read this whole document twice, found many logical inconsistencies, and corrected my English.

Many of my collegues deserve my thanks for helpful discussions, in particular Ralf Klinkenberg, Stefan Rüping, and Peter Brockhausen from the AI Unit. With the DFG Collaborative Research Center on Complexity Reduction in Multivariate Data (SFB475) supporting this work, I got much help from Thomas Fender, Ursula Sondhauß, Prof. Gather, and Prof. Weihs from the statistics department. While more difficult due to the distance, discussions and collaboration with Dr. Tobias Scheffer, Prof. Sebastian Thrun, Dr. Andrew McCallum, Dr. Dunja Mladenić, Marko Grobelnik, Dr. Justin Boyan, Dr. John Platt, Dr. Susan Dumais, Dr. Dayne Freitag, Dr. Mehran Sahami, Prof. Mark Craven, Prof. Lyle Ungar, Dr. Alex Smola, John Langford, Sean Slattery, and Rosie Jones substantially influenced this work. Of particular inspiration were the discussions with Prof. Vladimir Vapnik during his visit in Dortmund.

Last but not least, I would like to thank my parents Bernhard Joachims and Hiltrud Joachims, as well as my sister Martina Joachims. They supported me throughout my whole life in every respect and made me interested in science and learning about the world — sorry, Mom, it turned out not to be medicine.

Notation

$\vec{x}_i$	input patterns
y_i	target values (classes)
X	feature space
n	number of training examples
k	number of test examples
N	dimensionality of the input space
$\mathcal{L}$	learner
$\mathcal{H}$	hypothesis space
h	hypothesis from the hypothesis space
$h_{\mathcal{L}}$	hypothesis that the learner $\mathcal{L}$ returns
$R(h)$	(expected) risk of the hypothesis h
$R_{emp}(h)$	empirical risk of the hypothesis h on a training sample
L	loss function
$L_{0/1}$	0/1-loss function

$\vec{w}$	weight vector of a hyperplane $< \vec{w}, b >$
b	constant offset (or threshold) of a hyperplane $< \vec{w}, b >$
δ	margin of a hyperplane
R	radius of a ball containing the data, usually approximated by max $\|\vec{x}\|_2$
α_i	Lagrange multiplier
$\vec{\alpha}$	vector of all Lagrange multipliers
ξ_i	slack variables
$(\vec{x}_1 \cdot \vec{x}_2)$	dot product between vectors $\vec{x}_1$ and $\vec{x}_2$
$\mathcal{K}$	Mercer kernel
Q	Hessian of the quadratic program

Err	error rate
Rec	recall
$Prec$	precision
F_β	Rijsbergen's F_β-measure
$PRBEP$	precision/Recall breakeven point
$PRAVG$	arithmetic average of precision and recall

$\vec{x}^T$	transpose of the vector $\vec{x}$
$\Re$	the set of real numbers
$\mathbf{N}$	the set of natural numbers
$\|X\|$	cardinality of set X
$abs(a)$	absolute value of a
$\|.\|_1$	L_1-norm , $\|\vec{x}\|_1 := \sum abs(x_i)$
$\|.\|_2$ or $\|.\|$	L_2-norm (Euclidean distance), $\|\vec{x}\| := \sqrt{(\vec{x} \cdot \vec{x})}$
$exp(a)$	2.7182818^a
ln	logarithm to base 2.7182818

Contents

Chapter 1

Introduction

With the rapid growth of the World Wide Web, the task of classifying natural language documents into a predefined set of semantic categories has become one of the key methods for organizing online information. This task is commonly referred to as text classification. It is a basic building block in a wide range of applications. For example, directories like Yahoo! categorize Web pages by topic, online newspapers customize themselves to a particular user's reading preferences, and routing agents at service hotlines forward incoming email to the appropriate expert by content. While it was possible in the past to have human indexers do the category assignments manually, the exponential growth of the number of online documents and the increased pace with which information needs to be distributed has created the need for automatic document classification.

The first step towards automatic document classification is a knowledge engineering approach. Experts handcraft classification rules that can classify new documents automatically. However, handcrafting classification rules is difficult and time-consuming [Hayes and Weinstein, 1990]. Therefore, in many situations this approach is still too inefficient and impractical. For example, when the number of categories is large, handcrafting a classification rule for each category can be prohibitively expensive. Even worse, when the classification rule is needed as part of a desktop application that is customized to a particular user, an expert knowledge engineer simply is not available. Difficulties also arise when the category definitions change over time. In this case, maintaining classification rules by hand can quickly become intractable.

A machine-learning approach to building text-classification rules can overcome these problems. Given a relatively small set of manually classified training documents, the problem of learning text-classification rules can be cast as a supervised learning problem. The learning algorithm is given access to the labeled training documents and produces a classification rule automatically. While this learning problem was already studied in the past, the motivation for the work in this dissertation is to provide the first learning approach that can efficiently, effectively, and provably solve the challenge of learning text classifiers from examples for a large and well-defined class of problems.

1.1 Challenges

The task of learning text classifiers poses a new combination of challenges for machine learning. It is characterized by the following properties:

Large Input Space: In text classification, the input to the learner consist of natural language. Natural language is expressive enough to describe many phenomena in the world — many even with multiple, equivalent forms. As formulated in linguistics, "the sentences of a language may be unlimited in number" ([Lyons, 1968], Section 4.2.2), but surely it is large. For example, it is unlikely that the current sentence has ever been formulated before. Therefore, text classification inherently deals with a large space of potential examples. Even strongly simplifying transformations, like ignoring ordering on the word level and representing a piece of text merely by a histogram of word frequencies, still leads to input spaces with 30,000 and more dimensions.

Little Training Data: For most learning algorithms, the required number of training examples to produce a sufficiently accurate classification rule scales with the dimensionality of the input space. In this case, the number of training examples to assure a good learning result vastly outnumbers what can be made available with reasonable effort. For a polynomial least squares approach, Fuhr formulates the rule of thumb that *"there should be at least 50-100 elements in the learning sample per component of the polynomial structure"* [Fuhr et al., 1994, page 188]. However, when learning text classifiers, one is usually faced with a paradoxical situation of having fewer training examples than dimensions in the feature space.

Noise: Most natural language documents contain language mistakes. Despite my best efforts, you will probably find spelling errors, typos, and ungrammatical sentences in this dissertation. In machine-learning terminology, this can be interpreted as noise. Furthermore, the process of generating training documents often produces mislabeled examples, also leading to noise.

Complex Learning Tasks: The classes of text-classification tasks are generally based on the semantic understanding of natural language by humans. For example, the classes might describe the topic of the text, the reading preferences of a particular user, or the importance of email messages. For none of theses tasks formal and operational definitions exist. The learner must able to approximate such complex concepts.

Computational Efficiency: Training classifiers in a high-dimensional input space with several thousands of training examples can be a computationally difficult problem for conventional learners. To get useful learning methods for text classification in practical applications, it is necessary to develop training algorithms that can handle, in particular, the large number of features efficiently.

Providing a solution to this challenging class of learning problems motivates the methods, the theory, and the algorithms developed in this dissertation. The approach taken here follows Vapnik's idea of maximum-margin separation, most prominently implemented in support vector machines (SVMs).

1.2 Goals

This dissertation presents a new machine-learning approach to the problem of learning text classifiers from examples. It is not primarily about methods, nor primarily about theory, nor primarily about algorithms. Rather, it aims to address all relevant aspects of this particular class of learning problems. Currently, there is no other approach that is computationally efficient, for which there is a well-founded learning theory that describes its mechanics with respect to text classification, and that performs well and robustly in practice. The approach presented in the following overcomes these problems of conventional learning methods. The main aspects are as follows:

METHODS: Applying text classification in the real world requires robust learning methods. This dissertation proposes Vapnik's support vector machine (SVM) as a new method for inductive and transductive learning of text classifiers. It develops a new method for model selection that allows SVMs to be applied with robust effectiveness and high efficiency.

THEORY: This dissertation develops a learning theory of text classification. The impact of this theory is threefold. First, from a practical point of view, the theory gives guidelines for when and how to use SVMs for text classification. Second, the theory puts the currently informal problem of text classification on a formal ground. This makes it accessable to formal analysis and guides the development of new methods. And third, the theory provides guarantees about the learning result.

ALGORITHMS: Methods are of little practical use without efficient algorithms. Therefore, this dissertation develops algorithms for training inductive and transductive SVMs efficiently even on large problems.

This dissertation is divided into three parts, each reflecting one of these aspects. Since this work addresses all three issues, it requires a mixture of techniques and concepts from several domains. It is located at the intersection of machine learning, statistics, mathematical programming, and information retrieval.

Besides its contribution to the application problem of learning text classifiers, this work develops some widely applicable machine-learning techniques. While I limit their discussion to the text-classification problem, both the particular techniques and the general approach taken here are not limited to text classification. On a meta level, this dissertation demonstrates a conceptual approach to understanding a class of learning problems that can also be transferred to other domains.

1.3 Overview and Structure of the Argument

The approach presented in this dissertation is based on the key insight that margin, the complexity measure used e.g. in support vector machines, is ideal for text classification. In particular, this dissertation shows how maximum-margin methods can overcome both the learning theoretical and the algorithmic problems of high-dimensional input spaces in text classification. With respect to learning theory, it explains how it is possible to overcome the overfitting problem. From the computational perspective, it explores algorithms that do not depend on the dimensionality of the feature space. And finally, it demonstrates the robustness and state-of-the-art performance of maximum-margin methods in multiple learning scenarios and on multiple learning tasks.

According to these three aspects, this dissertation is structured into three parts – namely THEORY, METHODS, and ALGORITHMS. This reflects the holistic approach taken in this work. While it is difficult to clearly separate these three aspects, the first part mainly deals with learning theory, while the second part focuses on practical methods and experimental results, and the third part develops efficient algorithms.

1.3.1 Theory

In the theory part, I first present a statistical learning model of text classification. None of the conventional methods have an appropriate model that explains why and when they will perform well on a particular text-classification task. While the models for some methods, like the naive Bayes classifier, are overly restrictive and inappropriate for text, others, like decision-tree learners, rely purely on empirical evidence. Their suitability for learning text classifiers is not well understood. To overcome this deficiency, Chapter 4 presents a statistical learning model that connects the statistical properties of text-classification tasks with the generalization performance of a support vector machine. The model explains how the maximum-margin approach can avoid the "curse of dimensionality" for text classification even without a feature-selection step that is essential for many conventional methods. This is the first model explaining when and why a particular learning method works well for text classification. The model identifies sufficient conditions of text-classification tasks that provably lead to low classification error.

The theoretical model is based on an intensional description of the text-classification task. It models prior knowledge about the learning task and does not require training data. When training data becomes available, it can replace the intensional model. With the training data, the question of whether the learning algorithm is suitable for the task in terms of generalization performance becomes a problem of error estimation. Chapter 5 develops a method to estimate the generalization error, as well as, for example, precision and recall of a support vector machine. Unlike brute-force cross-validation and bootstrapping methods, the estimator is computationally very efficient and does not require additional data like hold-out testing. The estimate makes it possible to efficiently detect when the accuracy of a learned classification rule is appropriate for the task at hand.

1.3.2 Methods

The methods section of this dissertation starts with Chapter 6. It empirically evaluates SVMs for text classification and proposes a two step process for model selection. Model selection is the problem of selecting between different representations and parameter settings for a learning task. The most appropriate document representation depends on the particular task. Chapter 6 shows how selection among multiple representations, preprocessing steps like stemming, stopword removal, and weighting schemes, as well as the setting of other learning parameters can be done efficiently and without need for expert interventions. The resulting SVM is compared to conventional methods. It is shown that SVMs substantially outperform existing methods.

Chapter 7 proposes a new framework to model many text classification and information retrieval tasks more appropriately. The new model is based on the transductive setting. In contrast to inductive learning, in the transductive setting the examples to be classified are part of the input to the learner. A typical transductive learning task is relevance feedback in information retrieval. Since the learner operates on a fixed document collection, all documents that will be classified using the learned classification rule are available to the learning algorithm. A typical inductive learning task is the routing of e-mails. The documents to be classified arrive one by one and are not available at training time. The maximum-margin approach presented in this dissertation offers the flexibility to model both scenarios. Each text-classification problem can therefore be modeled in the most appropriate way, exploiting all information that is available.

1.3.3 Algorithms

The third part of this dissertation presents new algorithms for training inductive and transductive support vector machines. Unlike most conventional learning algorithms, their time complexity does not necessarily depend on of the dimensionality of the feature space. This is very important for text classification, since the feature space often contains more than 30,000 attributes.

But not only is the number of features generally high, the training set can be large as well. While training an inductive support vector machine can be reduced to a standard quadratic optimization problem, standard methods for solving such problems are inappropriate or even intractable for large data sets. Chapter 8 proposes and analyzes a new training algorithm for inductive SVMs that can handle large training sets with 50,000 and more examples efficiently. It also describes its implementation in SVM^{light} , software which has already found use in many scientific and commercial applications.

Finally, a solution to training transductive SVMs is presented in Chapter 9. Training a support vector machine in the transductive framework requires solving a mixed integer quadratic program for which there is no known efficient solution. Chapter 9 proposes and evaluates an algorithm that efficiently finds an approximation to the solution and can handle large data sets.

1.4 Summary

In summary, the thesis of this research is that taking a maximum-margin approach, I can provide the first framework for the problem of learning text classifiers from examples that combines a learning theoretical foundation, with methods giving state-of-the-art prediction performance, and efficient training algorithms. Developing the theoretical learning model of text classification, designing and evaluating practical methods for text classification based on support vector machines, and designing efficient training algorithms for large problems are the central goals of the work presented in this dissertation.

The following two chapters summarize the state-of-the-art that served as the starting point for this thesis.

Chapter 2

Text Classification

This chapter reviews the state-of-the-art in learning text classifiers from examples. First, it gives formal definitions of the most common scenarios in text classification — namely binary, multi-class, and multi-label. Furthermore, it gives an overview of different representations of text, feature selection methods, and criteria for evaluating predictive performance. The chapter ends with a description of the experimental setup used throughout this dissertation.

2.1 Learning Task

How precisely is the task of learning text classifiers from examples defined? To develop effective methods and to evaluate their results, it is necessary to define the learning task in a formal way. The following describes the common inductive learning setting that is used in most existing studies. The novel setting of transductive inference is introduced in Chapter 7 and is not considered here.

The goal in inductive text classification is to infer a classification rule from a sample of labeled training documents so that it classifies new examples with high accuracy. More formally, the learner $\mathcal{L}$ is given a training sample S of n examples

$$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n) \tag{2.1}$$

drawn independently and identically distributed (i.i.d.)[1] according to an unknown but fixed distribution $\Pr(\vec{x}, y)$. This distribution specifies the particular learning task. Each example consists of the document vector $\vec{x}$ and the class label y. The document vector $\vec{x}$ describes the documents. Most commonly, $\vec{x}$ is a high-dimensional vector describing which words occur in the document. Other document representations are discussed in Section 2.2. The form of the class label depends on the type of classification task. The following discusses binary, multi-class, and multi-label tasks individually. Each implies a particular measure of performance $R(h)$ – called risk – that measures how well a classification rule h performs. This performance measure is based on a loss function $L(h(\vec{x}), y) \in \Re$ that measures in how far the class label predicted by a classification rule $h(\vec{x})$ and the observed label y are different. The corresponding performance measure $R(h)$ is the expectation of the loss with respect to $\Pr(\vec{x}, y)$.

$$R(h) = \int L(h(\vec{x}), y) d \Pr(\vec{x}, y) \tag{2.2}$$

[1] While the i.i.d. assumption is essential for the theortical results in statistical learning theory, it is not essential for the learning algorithms presented in the following. They will work even if the i.i.d. assumption is (mildly) violated.

Since this performance measure depends on the unknown distribution $\Pr(\vec{x}, y)$, it cannot be computed directly. The only explicit information about $\Pr(\vec{x}, y)$ is the training sample S. Using this training sample S the learner $\mathcal{L}$ aims to find a classification rule $h_{\mathcal{L}} = \mathcal{L}(S)$ that minimizes the risk.

So far, both the loss function $L(h(\vec{x}), y)$ and the learning method $\mathcal{L}$ are not specified. Different loss function are discussed in the following, while Section 2.5 and Chapter 3 discuss learning algorithms.

2.1.1 Binary Setting

The binary setting is the simplest, yet most important formulation of the learning problem. The more complex tasks discussed in the following can be reduced to this case under mild assumptions. In the binary setting there are exactly two classes. For example, these two classes can be "relevant" and "non-relevant" in an information retrieval application. Similarly, a simple email filter may have the task to decide between "spam" and "non-spam" messages. This implies that the class label y has only two possible values. For notational convenience let these values be $+1$ and -1. So $y \in \{-1, +1\}$.

The most common loss function for the binary case is the 0/1-loss. This loss function checks whether the predicted class label $h(\vec{x})$ and y are equal. If $h(\vec{x})$ equals y, then it returns the value 0. If they are unequal, it returns 1.

$$L_{0/1}(h(\vec{x}), y) = \begin{cases} 0 & h(\vec{x}) = y \\ 1 & otherwise \end{cases} \tag{2.3}$$

The performance measure associated with this loss function is error rate $Err(h)$. Error rate is the probability of making a false prediction on an example that is randomly drawn according to $\Pr(\vec{x}, y)$.

$$Err(h) = \Pr(h(\vec{x}) \neq y | h) = \int L_{0/1}(h(\vec{x}), y) d\Pr(\vec{x}, y) \tag{2.4}$$

Error rate treats all types of errors equally. For example, in spam filtering it considers deleting an important message as spam just as bad as passing a spam message to the user. Clearly, this is inappropriate. The first type of error should incur a larger loss than the second type of error. This can be achieved using cost factors C_{-+} and C_{+-}. For the first type of error the loss function returns C_{-+} while it returns C_{+-} for the second type of error.

$$L_{0/1}(h(\vec{x}), y) = \begin{cases} C_{+-} & h(\vec{x}) = +1 \text{ and } y = -1 \\ C_{-+} & h(\vec{x}) = -1 \text{ and } y = +1 \\ 0 & otherwise \end{cases} \tag{2.5}$$

An SVM that minimizes misclassification cost instead of error rate is discussed in Section 3.4.

While error rate and misclassification cost are the quantities minimized by learning algorithms, the performance of a text classification rule is usually measured differently in practice. The most common performance measures are precision and recall, as well as combined measures like F_1 and the precision/recall breakeven point. These measures are introduced in Section 2.6. Ideally, one would like to optimize, for example, F_1 directly [Lewis, 1995]. Unfortunately, it is not clear how to design efficient learning algorithms for doing that. Therefore, error rate or misclassification cost is used as a proxy. A weak justification is that an error rate of 0 implies perfect precision and recall. Nevertheless, low error rate does not necessarily imply both high precision and high recall. However, the minimization of error rate or cost has proven successful in practice.

Most learning algorithms produce classification rules $h_{\mathcal{L}}(\vec{x})$ that not only output the binary classification +1 or -1, but also output a real number. This number is often related to $\Pr(y = +1|\vec{x})$, the probability that example $\vec{x}$ has the observed class label $y = +1$. If $h_{\mathcal{L}}(\vec{x})$ induces the same ordering as $\Pr(y = +1|\vec{x})$, the resulting precision/recall curves are optimal [Robertson, 1977]. The analysis in [Platt, 1999b] shows that the output of an SVM classification rule does produce a good ordering. It is comparable to the estimates of other methods that approximate $\Pr(y = +1|\vec{x})$ directly.

2.1.2 Multi-Class Setting

Some classification tasks involve more than two classes. For example, an email routing agent at a service hotline might need to forward an incoming message to one out of ten customer representatives. This means the class label y can assume 10, or in general l, different values. So, without loss of generality, $y \in \{1, \ldots, l\}$. Since 0/1-loss naturally generalizes to this multi-class setting, the previous definition of error rate also applies to multi-class classification. Similarly, cost factors can be introduced in a straightforward way.

Some text classification algorithms, like decision tree learners, can handle multi-class problems directly. While an approach to multi-class classification with SVMs exists [Weston and Watkins, 1998], it is computationally inefficient. Other learning algorithms are strictly limited to binary problems. However, a multi-class problem can always be split into a set of l binary tasks, if the binary learning algorithm provides an estimate of $\Pr(y = i|\vec{x})$. For each class i, a binary learning problem is generated as follows. For the i-th binary learning task, the class label is $y_{bin}^{(i)} = +1$ iff $y = i$. For $y \neq i$ the binary class label is $y_{bin}^{(i)} = -1$. Now an individual classifier is trained for each binary problem resulting in l binary classification rules $h^{(1)}, \ldots, h^{(l)}$. To classify a new example $\vec{x}$, the output of each $h^{(i)}(\vec{x})$ as an estimate of $\Pr(y = i|\vec{x})$ is analyzed. The example is classified into that class for which the corresponding $h^{(i)}(\vec{x})$ is largest.

A justification for this procedure gives Bayes's rule. Bayes's rule describes the optimal behavior of a classification algorithm that has access to $\Pr(y = i|\vec{x})$ for all classes i. Bayes showed that the error rate is minimized when the algorithm assigns each example to the class i for which $\Pr(y = i|\vec{x})$ is highest. Therefore, if the binary learning algorithm provides a good estimate of $\Pr(y = i|\vec{x})$, the resulting error rate will be low.

This reduction of a multi-class problem into l binary tasks is often called a *one-against-the-rest* strategy. A less-frequently used but promising alternative is *pair-wise* classification (e.g. [Kreßel, 1999]). This strategy leads to $l(l-1)/2$ binary classification problems. For each pair of classes, a learner is trained to discriminate between examples of just these two classes. New documents are classified by, for example, majority vote of all $l(l-1)/2$ predictions. A more efficient decision strategy is proposed in [Platt et al., 2000].

2.1.3 Multi-Label Setting

Most text-classification tasks fall into the multi-label setting. Unlike in the multi-class case, there is no one-to-one correspondence between class and document. Instead, for a fixed number l of categories, each document can be in multiple, exactly one, or no category at all. For example, these categories can be semantic topic identifiers for indexing news articles. Accordingly, a single news story could be in the categories SOCCER and GERMANY.

This setting can be modeled using a multivariate class label in the form of an l-dimensional binary vector, i.e. $\vec{y} \in \{+1, -1\}^l$. Each individual component indicates whether a document belongs to that category on not. Since the class label is an l-dimensional binary vector, the corresponding classification rule must output an l-dimensional binary vector as well.

No currently used text-classification algorithms can handle this multivariate y directly. Furthermore, it is not clear how to count errors in the multi-label setting. In the strict sense of 0/1-loss, an error occurs whenever the binary vector predicted by the classification rule is different from the observed class label vector $\vec{y}$. However, this loss function does not model "close misses". A good loss function for the multi-label setting should indicate in how many positions the prediction and the observed class label differ. A reasonable distance metric for binary vectors is Hamming distance. The Hamming distance counts the number of mismatches and results in the following loss function.

$$L_{Hamming}(h(\vec{x}), \vec{y}) = \sum_{i=1}^{l} L_{0/1}(h^{(i)}(\vec{x}), y^{(i)}) \tag{2.6}$$

$h^{(i)}(\vec{x})$ and $y^{(i)}$ denote the individual components of the binary vectors. So the expected loss equals the sum of the error rates of l binary tasks.

$$R(h) \quad = \quad \sum_{i=1}^{l} Err(h^{(i)}) \tag{2.7}$$

This motivates that a multi-label task can also be split up into a set of binary classification tasks. Each category is treated as a separate binary classification problem. Such a binary problem answers the question of whether or not a document should be assigned to a particular category. For each category i, define the binary classification task as follows. The class label is $y_{bin}^{(i)} = 1$ iff $y^{(i)} = 1$. For $y^{(i)} = -1$ the binary class label is $y_{bin}^{(i)} = -1$. Now an individual classifier is trained for each binary problem resulting in l binary classification rules $h^{(1)}, \ldots, h^{(1)}$. A category i is assigned to a document $\vec{x}$, if the corresponding classification rule $h^{(1)}(\vec{x})$ predicts $+1$.

Why and when is this split-up into binary tasks justified? The learning task in the multi-label setting is $\Pr(\vec{x}, y) = \Pr(\vec{x}, y^{(1)}, \ldots, y^{(l)})$. Again, Bayes's rule says that the optimal classification regarding $L_{0/1}$ is achieved for $y^{(1)}, \ldots, y^{(l)}$ maximizing $\Pr(y^{(1)}, \ldots, y^{(l)}|\vec{x})$. Assume that the categories are independent given the document vector. This is a rather mild assumption, since the document vector $\vec{x}$ should carry most of the relevant information. Then it holds that $\Pr(y^{(1)}, \ldots, y^{(l)}|\vec{x}) = \Pr(y^{(1)}|\vec{x}) \cdots \Pr(y^{(l)}|\vec{x})$. So, if each binary classifier maximizes $\Pr(y^{(i)}|\vec{x})$ individually, this will result in the maximum of $\Pr(y^{(1)}, \ldots, y^{(l)}|\vec{x})$. This shows that, under the independence assumption, minimizing error rate on each binary task leads to a minimum overall risk. It is an open question whether exploiting dependencies between categories can improve performance in practice.

2.2 Representing Text

The representation of example vectors $\vec{x}$ has a crucial influence on how well the learning algorithm can generalize. It has to "fit" to the implicit assumptions of the learning algorithm. Since these assumptions, like the form of classification rules and the preference ordering among them, are usually inherent to the learning algorithm – and therefore fixed – choosing the representation is the central modeling tool. Even text that is already stored in machine readable form (e.g. HTML, PDF, PostScript) is generally not directly suitable for most learning algorithms. It has to be transformed into the representation that is appropriate for both the learning algorithm and the classification task.

A fundamental problem when dealing with natural language is that context has a substantial influence on the meaning of a piece of text (see [Blair, 1992]). For example, the same word can have different meaning in two sentences. The word “bank” can refer to the financial institution, a piece of furniture, and the side of the river (synonyms). The same sentence can have different meanings depending on the speaker, the audience, and the situation. Different approaches to representing text for text classification recognize or ignore these dependencies to a varying extent. They can be structured according to the level on which they analyze text, according to the following categories:

1. Morphological Level: the structure of words
2. Lexical Level: complete words
3. Syntactic Level: the structure of sentences
4. Semantic Level: the meaning of text
5. Pragmatic Level: the meaning of text with respect to context and situation (e.g. dialog structure)

The basic building blocks on each level will be called *indexing terms*. So, on the lexical level, indexing terms refer to words, while on the syntactic level indexing terms can refer to phrases or whole sentences.

Structuring natural-language processing and analysis along these categories was found useful in linguistics. Nevertheless, no category can be treated independently of the others. On each level, ambiguities exist that can only be solved using the next higher level. On the lexical level alone, for example, it is not possible to decide whether the word “book” is a verb or a noun. This ambiguity can be resolved on the syntactic level (”Please book that flight!”).

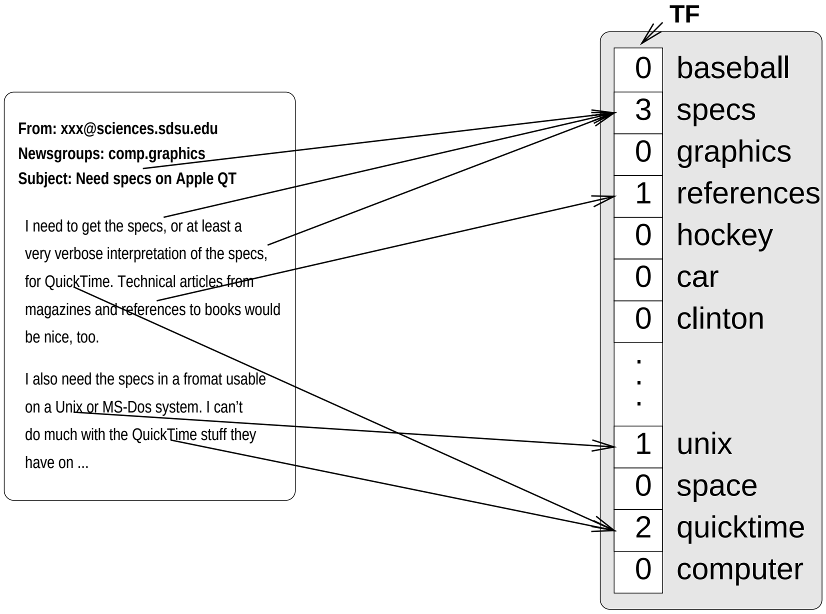
The following presents different representations based on the morphological, the lexical, the syntactic, and the semantic level for the English language. Representations recognizing the pragmatic structure of text have not been explored yet. Generally, the higher the level, the more detail the representation captures about the text. However, along with greater detail comes an increased complexity in producing such representations automatically. For example, producing semantic representations requires substantial knowledge about the domain of interest and can only be solved approximately with state-of-the-art methods.

The discussion begins with word-based representations. They are by far the most common way to represent text for text classification and will be used throughout this dissertation.

2.2.1 Lexical Level

In many cases words are meaningful units of little ambiguity even without considering context. While synonyms like “bank” exist, it is often assumed that they have little impact on the representation of a document as a whole. In fact, word-based representations have been found very effective in information retrieval and text classification (see e.g. [Lewis, 1992a]). They are the basis for most work in text classification.

A substantial advantage of word-based representations is their simplicity. It is relatively straightforward to design algorithms that efficiently decompose text into words. A



From: xxx@sciences.sdsu.edu
Newsgroups: comp.graphics
Subject: Need specs on Apple QT

I need to get the specs, or at least a very verbose interpretation of the specs, for QuickTime. Technical articles from magazines and references to books would be nice, too.

I also need the specs in a fromat usable on a Unix or MS-Dos system. I can't do much with the QuickTime stuff they have on ...

Figure 2.1: Representing text as an attribute vector.

simple algorithm that splits a string into words by white space characters will usually produce satisfactory results for the English language[2].

Ignoring logical structure and layout, using words as indexing terms transforms a document from a string of characters into a sequence of words. In addition, it is usually assumed that the ordering of the words is irrelevant (or at least of minor importance) for the classification task. So, a word sequence can be projected onto a bag of words. Only the frequency of a word in a document is recorded, while all structure of the document is ignored. This representation is commonly called the *bag-of-words* approach.

The bag-of-words representation makes text accessible to most machine-learning algorithms. These algorithms require that each example be described by a vector of fixed dimensionality. Each component of the vector represents the value of one attribute of the example. Commonly, each word w is treated as one such attribute. The value of an attribute for a particular document d can be, for example, the number of times it occurs in the document. This quantity is called the *term frequency* $TF(w,d)$ of word w in document d. Figure 2.1 illustrates how an example document is projected onto an attribute vector in the bag-of-words representation. Other methods for computing the value of an attribute are discussed in Section 2.4.

Representing documents as bags of words is a common technique in information retrieval. Clearly, this transformation leads to a loss of information about the document. However, more sophisticated representations have not yet shown consistent and substantial improvements. For information retrieval, Lewis formulated the following ([Lewis, 1992b], page 7):

> Reviews of this research have led to the widely-held conclusion that no text representation is significantly superior to representing documents by isolated words drawn from the original text, and that text representation has a relatively minor influence on effectiveness.

[2] In languages with composite nouns, finding individual words is more difficult. In Turkish even whole sentences can consist of a single composite expression.

While more expressive representations can capture more of the meaning of the document, their increased complexity degrades the quality of statistical models based on them. The bag-of-words representation appears to be a good compromise between expressiveness and model complexity.

The following hypothesis offers an intuitive explanation for why words are good representational units. The reason why words provide a reasonable granularity for representing documents might be found in the evolution of language. Words are the elements of the language where syntax and semantics meet. They are the basic syntactic building blocks that carry their own meaning. The vocabulary of a language is under constant development. Intuitively speaking, the composition and use of words is permanently optimized so that they encode an optimum of information. Clearly, such an optimum is relative to the tasks for which we use language [Whorf, 1959]. Language continuously adapts to the distribution of tasks, in particular by introducing new words. So the vocabulary of the language reflects what we deem as important. The same notion of importance also drives the text-classification tasks we usually consider and care about. The following example illustrates this. For an Eskimo, classifying weather reports by the type of expected snowfall might be a reasonable and important text-classification task. The Eskimo language is said to have many words differentiating between different types of snow [Whorf, 1959, page 216][3]. A bag-of-words approach will most likely be successful. However, the bag-of-words approach will probably fail using the English language on this task. English does not provide particular words, but requires more complicated constructions. On the other hand, it is very unlikely that someone from middle-europe will ever ask this text classification question[4]. In short, the hypothesis can be stated as follows:

> The vocabulary of a language reflects the prior distribution of text-classification tasks. Those text-classification tasks for which the vocabulary contains indicative keywords are a priori more likely.

Here, language does not only refer to ethnical languages, but also, for example, to technical languages of scientific disciplines. The validation of this hypothesis is an interesting open question.

2.2.2 Morphological Level

n-Grams are the most popular representation on the morphological level. Instead of using words as indexing terms, strings of n characters are here used as the basic building blocks. For example, the 3-gram (i.e. trigram) representation of the word "book" is "_bo", "boo", "ook", "ok_". The document as a whole is again a bag of these basic building blocks. The trigram representation naturally models similarity between words. While "computer" and "computers" are different words, they share most of their trigrams. While this is a desirable similarity, trigram based similarity can also be misleading ("computer" and "commuter"). An advantage of n-gram representations is that they provide some robustness against spelling errors. Furthermore, generating the n-gram representation of a document is straightforward even for languages with many composite forms. Experimental results can be found in e.g. [Neumann and Schmeier, 1999].

[3] Note that this example is still subject to debate, e.g. the necessity to include coding length.

[4] However, e.g. the German language does provide a rich vocabulary to describe different types of sausage and beer.

2.2.3 Syntactic Level

With improved tools linguistic tools, large amounts of text can now be analyzed efficiently with respect to their syntactic structure. Syntactic representations generally use indexing terms that consist of multiple words. Their composition is based on syntactic structure. The most commonly used syntactic structures are noun phrases (e.g. [Fagan, 1987][Croft and Lewis, 1990][Lewis, 1992b][Lewis, 1992a]). This approach is commonly called *Syntactic Phrase Indexing*. Other phrasal and syntactic features are explored in [Riloff and Lehnert, 1994][Fürnkranz et al., 1998][Neumann and Schmeier, 1999][Basili et al., 1999].

Another approach to generating multi-word indexing terms is based on statistical methods. Here, co-occurrence patterns are analyzed. A group of words is considered an indexing term if they co-occur frequently (e.g. [Fagan, 1987][Croft and Lewis, 1990]) or if they have a similar distribution with respect to the target concept [Baker and McCallum, 1998]. Fuhr et al. use syntactic preprocessing to generate candidates for indexing terms from which they filter good phrases using statistical methods [Fuhr et al., 1991].

2.2.4 Semantic Level

Clearly, text classifiers can only work optimally if they can capture the semantics of documents sufficiently. Unfortunately, it is not yet possible to automatically extract the semantics of free text and represent it in an operational form.

To some extent, the semantics of a document can be captured using taxonomies and indexing languages with a fixed vocabulary. Examples are MeSH for the field of medicine, the Dewey decimal classification for libraries, or Yahoo! categories for the World Wide Web. Documents are manually assigned to one or multiple categories. A hierarchical structure of the categories allows inferences along an is-a relationship. However, such an indexing of documents is usually not available. While the assignment of categories can potentially be automated [Yang and Chute, 1993], this task is itself a text-classification problem.

Section 2.3.2 discusses *Latent Semantic Indexing* [Deerwester et al., 1990]. Using a linear Principal-Component Analysis, this method aims to automatically generate semantic categories based on a bag-of-words representation. However, it is unclear why such a linear transformation should necessarily align meaning along orthogonal dimensions.

Predicate logic and semantic nets [Brachman and Schmolze, 1985] offer powerful representation languages that are very operational. However, representing documents in logic requires manual translation, which makes it impractical for most applications.

2.3 Feature Selection

Modifying the basic representation in a preprocessing step before running the learner is common practice in text classification. This step is called feature selection. Feature selection is supposed to remove irrelevant or inappropriate attributes from the representation. Protection against overfitting is the most common motivation for feature selection in text classification. A second motivation for feature selection is improving computational efficiency, since many learning algorithms cannot handle high-dimensional feature spaces. Some applications might impose strict memory and time constraints that can only be fulfilled using feature selection. This section reviews the most commonly used feature-selection techniques, differentiating between the following two approaches.

FEATURE SUBSET SELECTION The new representation consists of a subset of the original attributes.

FEATURE CONSTRUCTION New features are introduced by combining original features.

The description of feature-selection methods will assume a bag-of-words representation. However, most methods can also be applied to other representations in a straightforward way.

2.3.1 Feature Subset Selection

One method for identifying irrelevant features that applies exclusively to text is *stopword elimination*. It is based on the assumption that words like "the" or "and" are irrelevant independent of the classification task. Such words are removed from the feature vector.

While stopword elimination removes mostly high frequency words from the attribute set, *document frequency thresholding* [Yang and Pedersen, 1997] removes particularly infrequent words. All words that occur in less than m documents of the training corpus are not considered as features. Varying m can dramatically reduce the number of features even for small values of m. Document frequency thresholding is based on the assumption that infrequent words are not informative or, at least, not influential due to their rare occurrences [Yang and Pedersen, 1997]. Apté and Damerau justify this selection methods based on the statistical properties of low-frequency words [Apté and Damerau, 1994]. They conjecture that parameter estimates for low-frequency terms are not reliable enough to contribute useful information.

Unlike stopword elimination and document frequency thresholding, the following feature selection methods analyze the class labels in the training data to remove irrelevant attributes. Approaches for selecting attributes based on search and cross-validation (e.g. [Caruana and Freitag, 1994]) provide promising methods, but were not applied to text classification yet. For efficiency reasons most feature selection methods for text classification are based on ranking terms according to some statistical measure of relevance.

Mutual information (or *information gain* e.g.[Cover and Thomas, 1991][Quinlan, 1986]) is one of the most common measures of relevance in machine learning in general. It measures the reduction of entropy provided by considering two random variables Y and W together instead of individually.

$$\begin{aligned} I(Y,W) &= H(Y) - H(Y|W) && (2.8) \\ &= \sum_{y\in\{-1,+1\}} \sum_{w\in\{0,1\}} \Pr(y,w)\frac{\Pr(y,w)}{\Pr(y)\Pr(w)} && (2.9) \end{aligned}$$

Here, the random variable Y indicates the class label assigned to a document. The random variable W describes whether a particular word occurs in the document. Entropy $H(X)$ is the measure of uncertainty in the random variable X. It measures the expected number of bits necessary to encode X. $I(W,Y)$ describes the information that word W contributes to encoding the class label Y independent of the other words in the document. The probabilities can be estimated from the training sample using maximum likelihood estimates, leading to an estimate of $I(W,Y)$. The terms with the highest (empirical) mutual information are selected as features. However, it is unknown how to a priori select the number of features appropriately. Applications of mutual information can be found in e.g. [Yang and Pedersen, 1997][Lewis and Ringuette, 1994][van Rijsbergen, 1977][Lewis, 1992a][Mladenić, 1998].

Odds ratio [van Rijsbergen et al., 1981] is an alternative ranking measure commonly used in information retrieval. If the probability distribution $\Pr(Y|W)$ is known, then $min\,[\Pr(y=1|w), \Pr(y=-1|w)]$ is the Bayes optimal error rate that can be achieved, if only word W can be observed. This optimal error rate is achieved by the classifier $sign\left[\log \frac{\Pr(y=1|w)}{\Pr(y=-1|w)}\right]$. So under an independence assumption

$$\log \frac{\Pr(y=1|w)}{\Pr(y=-1|w)} \tag{2.10}$$

is a suitable measure of relevance for W. The probabilities are commonly estimated from the training set using Bayesian estimates. Experiments using odds ratio can be found in e.g. [Mladenić, 1998].

χ^2 tests (see e.g. [Mood et al., 1974]) are another way to measure the dependence between a term W and the class label Y. It tests W and Y for independence. The value of the test statistic is used as the ranking criterion. Experiments can be found in e.g. [Yang and Pedersen, 1997][Schűtze et al., 1995].

Note that feature selection based on such rankings implies strong assumptions about the learning task. It is a greedy process that does not account for dependencies between words. All statistical ranking criteria make independence assumptions of some sort. This dissertation shows that such a feature selection step is not necessary. This does not mean that selecting an appropriate representation is not crucial for solving a learning task. However, selecting attributes based on *training data* to improve generalization performance merely covers up deficiencies of the learning algorithm. Such feature selection uses the same information that is available to the learning algorithm – namely the training data. An appropriate learning algorithm should be able to detect irrelevant features as part of the learning process.

2.3.2 Feature Construction

The methods described in this section aim to reduce dimensionality by introducing new features. These new features should represents most of the information from the original representation while minimizing the number of attributes.

Stemming conducts a morphological analysis of words. Feature construction by stemming assumes that different word forms based on the same stem are equivalent with respect to the classification task. It collapses all words to their stem, introducing the stem as a new feature and removing features corresponding to word forms. This means that words like “computing”, “computability”, and “computer” are projected onto the same attribute “comput”. A simple rule-based algorithm for stemming English words is described in [Porter, 1980].

While stemming abstracts from the syntactical form of a word, *thesauri* (e.g. WordNet [Miller et al., 1990]) group words into semantic categories. A thesaurus can contain several kinds of relations between words. Synonyms, for example, are grouped into equivalence classes. Often thesauri also contain relations like more-general and more-specific. The use of thesauri for text classification is explored in e.g. [Junker and Abecker, 1997][Fisher, 1994][Scott and Matwin, 1998][de Buenaga Rodríguez et al., 1997].

Latent semantic indexing (LSI) [Deerwester et al., 1990] is a special form of linear principal component analysis (e.g. [Schölkopf, 1997]) applied to text. LSI produces a mapping of the feature vectors into a lower dimensional sub-space using singular value decomposition. It computes an orthogonal transformation of the coordinate system. The new coordinate values correspond to the new features. Keeping only the s largest singular values, the new feature space is of lower rank while preserving as much of the original feature vectors as possible. The best value of s is a priori unknown and has to be determined, for example, by cross-validation. The hope behind this transformation is that related words are clustered into the same principal component. Applications of latent semantic indexing for text classification can be found in [Foltz, 1990][Berry et al., 1995][Foltz and Dumais, 1992][Wiener et al., 1995]. LSI works exclusively on the feature vectors, but ignores class labels. Extensions to incorporate class information are proposed in [Wiener et al., 1995][Dumais, 1994][Schűtze et al., 1995][Yang, 1995].

Term clustering has a goal similar to that of LSI. Semantically similar terms shall be grouped into one cluster. The cluster then acts as a new feature. Clusters are generated using unsupervised learning algorithms (e.g. [Cheeseman et al., 1988]). These methods require that words are described using meta-attributes. The most common way of generating such meta-attributes for terms is by reversing the roles of documents and words. A word is described by the documents it occurs in (see e.g. [Spark-Jones, 1973][Croft and Lewis, 1990][Lewis, 1992a][Crouch, 1988]). Therefore, in a collection of 1000 documents, each term has 1000 meta-attributes. The clustering algorithms group words according to a distance measure among meta-attribute vectors. The hope is that this distance measure reflects semantic closeness by exploiting co-occurrence patterns.

2.4 Term Weighting

Intuitively, term weighting is a "soft" form of feature selection. While feature selection fully removes certain dimensions of the data, term weighting merely adjusts the relative influence of attributes. The term-weighting methods presented in the following were developed for the vector space model [Salton, 1971][Salton, 1991]. While originally developed for information retrieval, they have proven to be useful also for text classification.

Term-weighting schemes usually consist of three components [Salton and Buckley, 1988]. The *document component* captures statistics about a particular term in a particular document. The basic measure is *term frequency* $TF(w_i, d_j)$. It is defined as the number of times words w_i occurs in document d_j. The idea is that terms occurring more often in a document are more relevant than infrequent terms. However, frequent terms may occur in almost any document of a collection. Such terms will not help discriminate between classes. The *collection component* is designed to assign lower weight to such terms. Its basic statistic is *document frequency* $DF(w_i)$, i.e. number of documents in which word w_i occurs at least once. If the document frequency is high, the weight of the term is reduced. And finally, documents can be of different length. A *normalization component* is supposed to adjust the weights so that small and large documents can be compared on the same scale.

Table 2.1 lists the most frequently used choices for each component. For the final feature vector $\vec{x}$, the value x_i for word w_i is computed by multiplying the three components. The first column of the table defines an abbreviation that allows specifying choices in a compact way. The following combinations will be used in this dissertation.

bxc This string refers to a simple *binary* representation. Each word occurring in the document has weight 1, while all other words have weight 0. The resulting weight vector is normalized to unit length.

$$x_i = \frac{OCC(w_i, d)}{\sqrt{\sum_j OCC(w_j, d)^2}} \tag{2.11}$$

$OCC(w_i, d)$ returns 1, if word w_i occurs in document d, otherwise 0.

txc This representation uses the raw *term frequencies (TF)*. Again, length is normalized

Document Component		
b	1.0	binary weight equal to 1 for terms present in the document (term frequency is ignored)
t	$TF(w_i, d)$	raw term frequency
n	$0.5 + 0.5\frac{TF(w_i,d)}{\max_j TF(w_j,d)}$	augmented normalized term frequency (the impact of high term frequencies vs. low term frequencies is reduced)
Collection Component		
x	1.0	ignore document frequency
t	$\log \frac{\lvert D\rvert}{DF(w_i)}$	inverse document frequency. $\lvert D\rvert$ is that total number of documents in the collection. (terms occurring in many documents receive lower weight)
n	$\log \frac{\lvert D\rvert - DF(w_i)}{DF(w_i)}$	probabilistic inverse collection frequency (again, terms occurring in many documents receive lower weight)
Normalization Component		
x	1.0	no normalization
c	$\frac{1}{\sqrt{\sum x_j^2}}$	normalize resulting vector to length 1 (i.e. L_2)
a	$\frac{1}{\sum x_j}$	normalize resulting vector in terms of L_1

Table 2.1: Common word weighting components mostly taken from [Salton and Buckley, 1988].

according to L_2.

$$x_i = \frac{TF(w_i, d)}{\sqrt{\sum_j TF(w_j, d)^2}} \tag{2.12}$$

tfc This is the popular *TFIDF* representation with Euclidean length normalization.

$$x_i = \frac{TF(w_i, d)\ \log\left(\frac{|D|}{DF(w_i)}\right)}{\sqrt{\sum_j \left[TF(w_j, d)\ \log\left(\frac{|D|}{DF(w_j)}\right)\right]^2}} \tag{2.13}$$

Further details can be found in [Salton and Buckley, 1988].

2.5 Conventional Learning Methods

Throughout this dissertation, support vector machines will be compared to four standard learning methods, all of which have shown good results on text categorization problems in previous studies. Each method represents a different machine-learning approach: generative modelling using a naive Bayes classifier, the Rocchio algorithm (the most popular learning method from information retrieval), an instance-based k-nearest-neighbor classifier, and the C4.5 decision tree/rule learner. The following introduces these methods and gives and overview of other approaches.

2.5.1 Naive Bayes Classifier

The idea of the naive Bayes classifier is to use a probabilistic model of text to estimate $\Pr(y|d)$, the probability that a document d is in class y. To make the estimation of the parameters of the model possible, rather strong assumptions are incorporated. In the following, word-based unigram models of text will be used (cf. [Joachims, 1997a]). This multinomial event model is different from the multivariate Bernoulli model used by Lewis [Lewis, 1992c] and earlier proposed by [Maron, 1961]. Unlike the multivariate Bernoulli model, it can take the occurrence frequency of a word in a document into account and it is found to have better classification accuracy [McCallum and Nigam, 1998].

In the multinomial mixture model, words are assumed to occur independently of the other words in the document. For each category, there is one component in the model. All documents assigned to a particular category are assumed to be generated according to the component associated with this category.

The following describes one approach to estimating $\Pr(y|d)$. Bayes's rule says that to achieve the highest classification accuracy, d should be assigned to the class $y \in \{-1, +1\}$ for which $\Pr(y|d)$ is highest.

$$h_{BAYES}(d) = argmax_{y \in \{-1,+1\}} \Pr(y|d) \tag{2.14}$$

$\Pr(y|d)$ can be split up by considering documents separately according to their length l.

$$\Pr(y|d) = \sum_{l=1}^{\infty} \Pr(y|d, l) \cdot \Pr(l|d) \tag{2.15}$$

$\Pr(l|d)$ equals one for the length l' of document d and is zero otherwise. After applying Bayes's theorem to $\Pr(y|d, l)$ we can therefore write:

$$\Pr(y|d) = \frac{\Pr(d|y, l') \cdot \Pr(y|l')}{\sum_{y' \in \{-1,+1\}} \Pr(d|y', l') \cdot \Pr(y'|l')} \tag{2.16}$$

Document (d):	*Several*	*methods*	*for*	*text*	*categorization*		*have*	*...*	
Category1 (y1):	0.012 *	0.007 *	0.023 *	0.035 *	0.021	*	0.040 *	...	= $\hat{P}(d \mid y1)$
Category2 (y2):	0.011 *	0.001 *	0.026 *	0.006 *	0.001	*	0.042 *	...	= $\hat{P}(d \mid y2)$
⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮

Figure 2.2: In the multinomial model a document is assumed to be generated by repeatedly drawing words i.i.d. from the mixture component corresponding to the class of the document.

$\Pr(d|y,l')$ is the probability of observing document d in class y given its length l'. $\Pr(y|l')$ is the prior probability that a document of length l' is in class y. In the following we will assume that the category of a document does not depend on its length, so $\Pr(y|l') = \Pr(y)$. An estimate $\hat{\Pr}(y)$ for $\Pr(y)$ can be calculated from the fraction of training documents that is assigned to class y.

$$\hat{\Pr}(y) = \frac{|y|}{\sum_{y' \in \{-1,+1\}} |y'|} = \frac{|y|}{|D|} \tag{2.17}$$

$|y|$ denotes the number of training documents in class $y \in \{-1,+1\}$ and $|D|$ is the total number of documents.

The estimation of $\Pr(d|y,l')$ is more difficult. $\Pr(d|y,l')$ is the probability of observing a document d in class y given that we consider only documents of length l'. Since there is - even for a simplifying representation as used here - a huge number of different documents, it is impossible to collect a sufficiently large number of training examples to estimate this probability without prior knowledge or further assumptions. In our case the estimation becomes possible due to the way documents are assumed to be generated. The unigram models introduced above imply that a word's occurrence is only dependent on the class the document comes from, but that it occurs independently[5] of the other words in the document and that it is not dependent on the document length. This is illustrated in Figure 2.2. So $\Pr(d|y,l')$ can be written as:

$$\Pr(d|y,l') \approx \prod_{i=1}^{|d|} \Pr(w_i|y) \tag{2.18}$$

w_i ranges over the sequence of words in document d which are considered as features. $|d|$ is the number of words in document d. The estimation of $\Pr(d|y)$ is reduced to estimating each $\Pr(w_i|y)$ independently. A Bayesian estimate is used for $\Pr(w_i|y)$.

$$\hat{\Pr}(w_i|y) = \frac{1 + TF(w_i,y)}{|F| + \sum_{w' \in |F|} TF(w',y)} \tag{2.19}$$

$TF(w,y)$ is the overall number of times word w occurs within the documents in class y. This estimator, which is often called the Laplace estimator, is suggested in [Vapnik, 1982, pages 54-55]. It assumes that the observation of each word is a priori equally likely.

[5] The weaker assumption of "linked-dependence" is actually sufficient [Cooper, 1991], but not considered here for simplicity.

It was found that this Bayesian estimator works well in practice, since it does not falsely estimate probabilities to be zero [Joachims, 1997a].

The following is the resulting classification rule if equations (2.14), (2.16) and (2.18) are combined.

$$h_{BAYES}(d) = argmax_{y \in \{-1,+1\}} \frac{\Pr(y) \cdot \prod_{i=1}^{|d|} \Pr(w_i|y)}{\sum_{y' \in \{-1,+1\}} \Pr(y') \cdot \prod_{i=1}^{|d|} \Pr(w_i|y')} \tag{2.20}$$

$$= argmax_{y \in \{-1,+1\}} \frac{\Pr(y) \cdot \prod_{w \in X} \Pr(w|y)^{TF(w,d)}}{\sum_{y' \in \{-1,+1\}} \Pr(y') \cdot \prod_{w \in X} \Pr(w|y')^{TF(w,d)}} \tag{2.21}$$

If $\Pr(y|d)$ is not needed as a measure of confidence, the denominator can be left out, since it does not change the argmax. Experimental results for naive Bayes classifiers can be found in e.g. [Lewis, 1992c][Lewis and Ringuette, 1994][Lang, 1995][Pazzani et al., 1996][Joachims, 1997a][Joachims, 1998b][McCallum and Nigam, 1998][Sahami, 1998].

2.5.2 Rocchio Algorithm

This type of classifier is based on the relevance-feedback algorithm originally proposed by Rocchio [Rocchio, 1971] for the vector-space retrieval model [Salton, 1991]. It has been extensively used for text classification.

First, both the normalized document vectors of the positive examples as well as those of the negative examples are summed up. The linear component of the classification rule is then computed as

$$\vec{w} = \frac{1}{|i : y_i = +1|} \sum_{i:y_i=+1} \vec{x}_i - \beta \frac{1}{|j : y_j = -1|} \sum_{j:y_j=+1} \vec{x}_j \tag{2.22}$$

Rocchio requires that negative elements of the vector w are set to 0. β is a parameter that adjusts the relative impact of positive and negative training examples. Buckley [Buckley et al., 1994] recommends $\beta = 0.25$. Nevertheless, the optimal values are likely to be task-dependent and the performance of the resulting classifier strongly depends on a "good" choice of β.

To classify a new document $\vec{x}$, the cosine between $\vec{w}$ and $\vec{x}$ is computed. Using an appropriate threshold on the cosine leads to a binary classification rule. The Rocchio algorithm does not provide a means to compute a good threshold. One solution is to obtain a threshold via hold-out testing.

Why should the weight vector $\vec{w}$ classify new documents well? With the cosine as the similarity metric and $\beta = 1$, Rocchio shows that for $\vec{w}$ as computed above, the mean similarity of the positive training examples with $\vec{w}$ minus the mean similarity of the negative training examples with $\vec{w}$ is maximized.

$$\frac{1}{|i : y_i = +1|} \sum_{i:y_i=+1} \cos(\vec{w}, \vec{x}_i) - \frac{1}{|j : y_i = -1|} \sum_{j:y_i=-1} \cos(\vec{w}, \vec{x}_j) \longrightarrow max \tag{2.23}$$

However, it is unclear if or how maximizing this function connects to the accuracy of the resulting classifier. The standard parametric distributions that imply good generalization performance appear inappropriate for text. Experiments with the Rocchio algorithm are reported in [Schűtze et al., 1995] [Balabanovic and Shoham, 1995] [Lang, 1995] [de Kroon et al., 1996] [Cohen, 1996] [Joachims, 1997a] [Joachims, 1998b] [Dumais et al., 1998].

2.5.3 k-Nearest Neighbors

The k-nearest-neighbor (k-NN) classifier is based on the assumption that examples located close to each other according to a user-defined similarity metric are likely to belong to the same class. Like the naive Bayes classifier, it can be derived from Bayes's rule [Michie et al., 1994]. k-NN classifiers were found to show good performance on text-categorization tasks [Yang, 1997][Yang, 1999] [Masand et al., 1992]. This paper follows the setup in [Yang, 1997]. The cosine is used as a similarity metric. $knn(\vec{x})$ denotes the indices of the k documents which have the highest cosine with the document to classify $\vec{x}$.

$$h_{knn}(\vec{x}) = sign\left(\frac{\sum\limits_{i \in knn(\vec{x})} y_i \cos(\vec{x}, \vec{x}_i)}{\sum\limits_{i \in knn(\vec{x})} \cos(\vec{x}, \vec{x}_i)}\right) \tag{2.24}$$

Further details can be found in [Mitchell, 1997].

2.5.4 Decision Tree Classifier

Representative of decision-tree learners, the C4.5 algorithm [Quinlan, 1993] is used for the experiments in this paper. It is the most popular decision-tree algorithm and has shown good results on a variety of problems. It is used with the default parameter settings and with rule post-pruning turned on. C4.5 outputs a confidence value when classifying new examples. This value is used to compute precision/recall tables. Previous results with decision tree or rule learning algorithms are reported in [Lewis and Ringuette, 1994] [Moulinier et al., 1996][Apté and Damerau, 1994][Cohen, 1995][Cohen, 1996].

2.5.5 Other Methods

One of the shortcomings of the naive Bayes classifier is its unjustified conditional independence assumption. It can be interpreted as a *Bayesian network* in its simplest form. Using more general Bayesian network models, researchers have tried to overcome the limitations of naive Bayes [Sahami, 1998][Tzeras and Hartmann, 1993]. Sahami was able to show that an automatically constructed Bayesian network with limited dependence can improve prediction performance.

Logistics regression (e.g. [Michie et al., 1994]) is a different way of estimating the probability $\Pr(y|\vec{x})$. Instead of estimating a generative model, logistic regression takes a discriminative approach. Similar to a linear support vector machines, statistic regression finds a hyperplane in feature space. The difference is that statistic regression optimizes conditional likelihood on the training data. Text-classification experiments with unregularized logistic regression can be found in [Schűtze et al., 1995]. However, regularization like in the SVM should improve performance also for logistic regression.

Neural nets (e.g. [Mitchell, 1997]) are closely related to logistic regression, but deal with more complex than linear models. Since neural nets are sensitive to overfitting, they require feature selection like latent semantic indexing. This combination is explored in [Schűtze et al., 1995] and [Wiener et al., 1995]. A similar approach using *linear regression* is explored in [Yang, 1999]. Linear regression is a special case of *polynomial regression* [Fuhr, 1989], used in pioneering work on text classification as part of the system AIR/PHYS [Fuhr and Knorz, 1984].

Boosting algorithms like AdaBoost [Freund and Schapire, 1996] iteratively combine multiple base hypotheses (e.g. decision trees) using a linear model. Boosted decision trees have shown excellent performance on text-classification tasks. The size of the trees can vary from minimal trees of depth 1 (i.e. decision stumps) [Schapire et al., 1998][Schapire and Singer, 2000] to large trees [Weiss et al., 1999]. Especially for boosted decision stumps there is a strong connection to support vector machines [Schapire et al., 1997], since Boosting can also be interpreted as margin maximization. Instead of measuring margin in terms of the L_2-norm like in the SVM, Boosting maximizes margin in terms of the L_1-norm. With a modified loss function, Boosting can be formulated as an optimization problem similar to that of the support vector machine (see Optimization Problem 3 in Section 3.2).

Rule learning approaches to text classification focus on good search strategies and compact representations. For example, Haneke explores genetic search [Haneke, 1999]. Often, rules and decision trees are found to be more interpretable than, for example, linear models. Lewis and Ringuette compare a decision tree learner with a naive Bayes classifier and find similar performance. Unlike C4.5, SWAP-1 [Weiss and Indurkhya, 1993] learns Horn clauses without building an intermediate decision tree. It can include consecutive word pairs as features, leading to good results on the Reuters corpus [Apté and Damerau, 1994]. Further experiments with several rule learning algorithms can be found in [Cohen, 1995][Moulinier et al., 1996][Moulinier and Ganascia, 1996].

Relational rule learning offers a more powerful representation than the propositional methods discussed so far. Relational predicates are an elegant way to express relations between attributes. For example, it is easy to represent word ordering, which is usually ignored in propositional representations. Cohen uses the following relations between words of a document [Cohen, 1995][Cohen, 1996]:

- *near1/2*: two words are adjacent
- *near2/2*: two words are separated by at most one word
- *near3/2*: two words are separated by at most two words
- *after/2*: one word occurs in the document after another word
- *near1/2*: one word directly follows another word

Cohen uses the FOIL6 [Quinlan and Cameron-Jones, 1993] and the FLIPPER [Cohen, 1995] ILP learning algorithms on this representation. Compared to propositional representations Cohen concludes that the relational approach can improve performance on some problems. However, the improvements are rather small so that other parameters have a larger impact on the result [Cohen, 1995].

Active learning is an interesting modification of the inductive learning setting. The learner does not observe training examples in a passive way, but can actively ask for e.g. the label of particular examples. This can reduce the total number of labeled training examples needed to achieve the same performance as a passive learner. Active learning for text classification was first explored in [Lewis and Gale, 1994]. The recent Ph.D. thesis of Ray Liere gives interesting results for Winnow and Perceptron learners [Liere, 1999]. Since Winnow and especially the Perceptron algorithm can be interpreted as "online versions" of support vector machines with particular norms, it is plausible that his results will transfer to SVMs.

2.6 Performance Measures

This section reviews the most commonly used performance measures for evaluating text classifiers. Like already discussed in Section 2.1.1, the performance measures used for evaluating text classifiers are often different from those optimized by the learning algorithm.

The following discusses loss-based measures (i.e. error rate and cost models), precision and recall, as well as combined measures like F_1 and the precision/recall breakeven point (PRBEP).

Estimators for these measures can be defined based on a contingency table of predictions on an independent test set. The following contingency table provides a convenient display of the prediction behavior for binary classification.

Definition 1 (Contingency Table (Binary Classification))

	label $y = +1$	*label* $y = -1$
pediction $h(\vec{x}) = +1$	f_{++}	f_{+-}
pediction $h(\vec{x}) = -1$	f_{-+}	f_{--}

Each cell of the table represents one of the four possible outcomes of a prediction $h(\vec{x})$ for an example $(\vec{x}, y)$. The diagonal cells count how often the prediction was correct. The off-diagonal entries show the frequency of prediction errors. The sum of all cells equals the total number of predictions.

2.6.1 Error Rate and Asymmetric Cost

The most common performance measure in machine learning is error rate. It is defined as the probability of the classification rule h predicting the wrong class.

Definition 2 (Error Rate)

$$Err(h) \quad = \quad \Pr(h(\vec{x}) \neq y | h) \tag{2.25}$$

From the contingency table it can be estimated as

$$Err_{test}(h) \quad = \quad \frac{f_{+-} + f_{-+}}{f_{++} + f_{+-} + f_{-+} + f_{--}} \tag{2.26}$$

However, unlike for other applications of machine learning, error rate alone is not necessarily a good performance measure in text classification. Usually negative examples vastly outnumber positive examples. So the simple classifier always responding $h(\vec{x}) = -1$ independent of the particular feature vector $\vec{x}$ has a (misleadingly) low error rate.

This behavior is due to the equal weighting of false positives and false negatives implicit in error rate. For many applications, predicting a positive example correctly is of higher utility than predicting a negative example correctly. It is possible to incorporate this into the performance measure using a cost (or, inversely, utility) matrix.

Definition 3 (Cost Matrix)

	label $y = +1$	*label* $y = -1$
pediction $h(\vec{x}) = +1$	C_{++}	C_{+-}
pediction $h(\vec{x}) = -1$	C_{-+}	C_{--}

The elements of the cost matrix are multiplied with the corresponding entries of the contingency table. They form a linear cost function.

$$Cost_{test}(h) \quad = \quad \frac{C_{++}\, f_{++} + C_{+-}\, f_{+-} + C_{-+}\, f_{-+} + C_{--}\, f_{--}}{f_{++} + f_{+-} + f_{-+} + f_{--}} \tag{2.27}$$

In particular, errors on positive examples and errors on negative examples can be weighted differently using C_{+-} and C_{-+}. A support vector machine with such a cost model is introduced in Section 3.4.

2.6.2 Precision and Recall

While cost models address the problem of unequal class distributions, their resulting score is often not intuitively interpretable. A more detailed summary give scores based on precision and recall. These measures are of widespread use in information retrieval and can also be found in ROC analysis [Provost and Fawcett, 1997]. I define the recall $Rec(h)$ of a classification rule h as the probability that a document with label $y = 1$ is classified correctly.

Definition 4 (Recall)

$$Rec(h) \quad = \quad \Pr(h(\vec{x}) = 1 | y = 1, h) \tag{2.28}$$

Based on the contingency table in Definition 1, a straightforward estimate of the recall is

$$Rec_{test}(h) \quad = \quad \frac{f_{++}}{f_{++} + f_{-+}} \tag{2.29}$$

The precision $Prec(h)$ of a classification rule h is the probability that a document classified as $h(\vec{x}) = 1$ is indeed classified correctly.

Definition 5 (Precision)

$$Prec(h) \quad = \quad \Pr(y = 1 | h(\vec{x}) = 1, h) \tag{2.30}$$

Similar to recall, a straightforward estimate of the precision is

$$Prec_{test}(h) \quad = \quad \frac{f_{++}}{f_{++} + f_{+-}} \tag{2.31}$$

Between high precision and high recall exists a trade-off. All methods examined in this dissertation make category assignments by thresholding a “confidence value”. By adjusting this threshold it is possible to achieve different levels of recall and precision. When the confidence value does not imply a total ordering, the PRR method [Raghavan et al., 1989] is used for interpolation.

2.6.3 Precision/Recall Breakeven Point and F_β-Measure

While precision and recall accurately describe the classification performance, considering two scores makes it difficult to compare different learning algorithms. To get a single performance measure, the (weighted) harmonic mean of precision and recall is commonly used. It is called the F_β-measure and can be written as follows.

Definition 6 (F_β-Measure)

$$F_\beta(h) = \frac{(1 + \beta^2)\; Prec(h)\; Rec(h)}{Prec(h) + Rec(h)} \tag{2.32}$$

β is a parameter. The most commonly used value is $\beta = 1$, giving equal weight to precision and recall. The F_β-measure can be estimated from the contingency table using

$$F_{\beta,test}(h) = \frac{(1 + \beta^2)\; f_{++}}{(1 + \beta^2)\; f_{++} + f_{-+} + \beta^2 f_{+-}} \tag{2.33}$$

Another measure that summarizes the precision/recall curve using a single value is the precision/recall breakeven point. It assumes that the classification rule returns a confidence value which ranks test examples according to how likely they are in the positive class.

Definition 7 (Precision/Recall Breakeven Point) *Given a ranking of documents, the precision/recall breakeven point (PRBEP) is the value at which precision and recall are equal.*

It is easy to verify that precision and recall are necessarily equal when the number of test examples predicted to be in the positive class equals the true number of positive test examples (i.e. $f_{++} + f_{+-} = f_{++} + f_{-+}$).

Clearly, summarizing the whole precision/recall curve using a single value leads to a loss of information. Both F_β and the PRBEP make assumptions about the relative importance of recall and precision. These assumptions may not be met for a particular application. However, even without a particular application in mind, they do provide useful scores for comparing learning algorithms.

2.6.4 Micro- and Macro-Averaging

Often it is useful to compute the average performance of a learning algorithm over multiple training/test sets or multiple classification tasks. In particular for the multi-label setting, one is usually interested in how well all the labels can be predicted, not only a single one. This leads to the question of how the results of m binary tasks can be averaged to get to a single performance value. There are two common approaches for computing such averages – macro-averaging and micro-averaging.

Macro-averaging corresponds to the standard way of computing an average. The performance measure (i.e. precision, recall, etc.) is computed separately for each of the m experiments. The average is computed as the arithmetic mean of the performance measure over all experiments. For the F_1-measure this implies

$$F_1^{macro} = \frac{1}{l} \sum_{i=1}^{m} F_1(h_i) \tag{2.34}$$

Micro-averaging does not average the resulting performance measure, but instead averages the contingency tables. For each cell of the table the arithmetic mean is computed, leading to an averaged contingency table with elements f_{++}^{avg}, f_{+-}^{avg}, f_{-+}^{avg}, and f_{--}^{avg}. Based on this table, the performance measure is computed. For the $F1$-measure this implies

$$F_1^{micro} = \frac{2\, f_{++}^{avg}}{2\, f_{++}^{avg} + f_{-+}^{avg} + f_{+-}^{avg}} \tag{2.35}$$

While computing the micro-average is straightforward also for precision and recall, the classification threshold is not fixed for the precision/recall breakeven point. To get the micro-average of the PRBEP the following procedure is used. The classification threshold Θ is lowered simultaneously over all binary tasks[6]. At each value of Θ the microaveraged precision and recall are computed based on the merged contingency table. To arrive at this merged table, the contingency tables of all binary tasks at Θ are added componentwise.

2.7 Experimental Setup

To give a clear idea about the impact of the theoretical results for text classification, all arguments in this dissertation are evaluated and verified in experiments. This section motivates and defines the experimental setup, namely the test collections, the choice of representation, and the evaluation criteria.

[6] Since cosine similarities are not comparable across classes, the method of *proportional assignment* [Wiener et al., 1995] is used for the Rocchio algorithm to come up with improved confidence values.

2.7.1 Test Collections

The empirical evaluation is done on three test collection. They were chosen to represent a wide spectrum of text-classification tasks.

The first one is the *Reuters-21578* dataset[7] compiled by David Lewis and originally collected by the Carnegie group from the Reuters newswire in 1987. The "ModApte" split is used, leading to a corpus of 9,603 training documents and 3,299 test documents. Of the 135 potential topic categories only those 90 are used for which there is at least one training and one test example. The Reuters-21578 collection is know to have a fairly restricted vocabulary. While there are many proper nouns, the training examples contain only 27,658 distinct terms that occur at least once. The articles are authored by professional writers according to guidelines. The classification task is to assign articles to a set of topics. For many topic categories there is a rather direct correspondence between words and categories. For the topic category "wheat" for example, the occurrence of the word "wheat" in a document is a very good predictor.

The second dataset is the *WebKB* collection[8] of WWW pages made available by the CMU text-learning group. Following the setup in [Nigam et al., 1998], only the classes `course`, `faculty`, `project`, and `student` are used. Documents not in one of these classes are deleted. After removing documents which just contain the relocation command for the browser, this leaves 4,183 examples. The pages from Cornell University are used for testing (229 examples), while all other pages are used for training (3957 examples). The type of text is very different from the Reuters collection. The documents are WWW pages with very heterogeneous writing styles, incomplete sentences, and structural information. After removing all HTML tagging, building a dictionary on the training examples leads to 38,359 distinct words. Another substantial difference is that the classification is not by topic, but by the function of the page.

The third test collection is taken from the *Ohsumed* corpus[9] compiled by William Hersh. From the 50,216 documents in 1991 which have abstracts, the first 10,000 are used for training and the second 10,000 are used for testing. The classification task considered here is to assign the documents to one or multiple categories of the 23 MeSH "diseases" categories. A document belongs to a category if it is indexed with at least one indexing term from that category. The documents are medical abstracts, strongly dominated by special medical terms. 38,679 distinct terms occur in the training examples once or several times. Unlike the Reuters articles, Ohsumed abstracts address a highly specialized and trained audience. At least for a person illiterate in medicine, the connection between words and categories is less direct than for Reuters.

2.7.2 Design Choices

The following lists the basic design decisions for the experiments in this dissertation. While most particular choices are unlikely to have a substantial impact on the qualitative results of the experiments, they are listed to make the experiments more understandable and reproduceable.

- Words are chosen as the basic representational units. This is the standard approach. However, richer representations (e.g. noun phrases) can easily be incorporated.
- Words are defined as non-whitespace strings enclosed by whitespace characters. All punctuation marks count as whitespace characters. Capital letters are transformed

[7] Available at http://www.research.att.com/~lewis/reuters21578.html
[8] Available at http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data
[9] Available at ftp://medir.ohsu.edu/pub/ohsumed

to lower-case.

- In the inductive setting, only those words occurring in the training data are considered as potential features. In the transductive setting, also the test documents contribute words and their document frequency is used for computing term weights.
- All numbers are projected onto a single token.
- Feature selection (e.g. stemming, stopword removal, etc.) is not used unless noted otherwise.
- Stemming is performed using the Porter algorithm [Porter, 1980] implemented by B. Frakes and C. Cox.
- Stopword removal is performed according to the FreeWAIS stoplist given in the appendix.
- To make short and long documents comparable, feature vectors are normalized to unit length (not naive Bayes and C4.5 using the binary vector representation).
- The one-against-the-rest approach is used to handle multi-label problems.
- To facilitate comparative studies it is necessary to have a single-valued measure of predictive performance. Any such measure makes assumptions about the relative importance of false positives and false negatives. Without a particular application (and associated costs) the choice is somewhat arbitrary. For the following reasons, the precision/recall breakeven point is used as the prime evaluation measures in this dissertation. The PRBEP provides a meaningful score that summarizes the precision/recall curve in an intuitive way. Unlike F_1, the PRBEP stands for a precision and recall actually achievable by the classification rule. The PRBEP is already widely used as an evaluation criterion, allowing a comparison with existing work.
- All significance tests are regarding a 95% confidence level.

Chapter 3

Support Vector Machines

This chapter gives a short introduction to support vector machines, the basic learning method used, extended, and analyzed for text classification throughout this dissertation. Support vector machines [Cortes and Vapnik, 1995][Vapnik, 1998] were developed by Vapnik et al. based on the *Structural Risk Minimization* principle [Vapnik, 1982] from statistical learning theory. The idea of structural risk minimization is to find a hypothesis h from a hypothesis space H for which one can guarantee the lowest probability of error $Err(h)$ for a given training sample S

$$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n) \qquad \vec{x}_i \in \Re^N, y_i \in \{-1, +1\} \tag{3.1}$$

of n examples. The following upper bound connects the true error of a hypothesis h with the error $Err_{train}(h)$ of h on the training set and the complexity of h [Vapnik, 1998] (see also Section 5.1.1).

$$Err(h) \leq Err_{train}(h) + O\left(\frac{d\ln(\frac{n}{d}) - \ln(\eta)}{n}\right) \tag{3.2}$$

The bound holds with a probability of at least $1 - \eta$. d denotes the *VC-dimension* [Vapnik, 1998], which is a property of the hypothesis space H and indicates its expressiveness. Equation (3.2) reflects the well-known trade-off between the complexity of the hypothesis space and the training error. A simple hypothesis space (small VC-dimension) will probably not contain good approximating functions and will lead to a high training (and true) error. On the other hand a too rich hypothesis space (large VC-dimension) will lead to a small training error, but the second term in the right-hand side of (3.2) will be large. This reflects the fact that for a hypothesis space with high VC-dimension the hypothesis with low training error may just happen to fit the training data without accurately predicting new examples. This situation is commonly called "overfitting". It is crucial to pick the hypothesis space with the "right" complexity.

In Structural Risk Minimization this is done by defining a nested structure of hypothesis spaces H_i, so that their respective VC-dimension d_i increases.

$$H_1 \subset H_2 \subset H_3 \subset \ldots \subset H_i \subset \ldots \qquad \text{and} \qquad \forall i : d_i \leq d_{i+1} \tag{3.3}$$

This structure has to be defined a priori, i.e. before analyzing the training data. The goal is to find the index i^* for which (3.2) is minimum.

How can one build this structure of increasing VC-dimension in practice? SVMs learn linear threshold functions of the type:

$$h(\vec{x}) = sign\{\vec{w} \cdot \vec{x} + b\} = \begin{cases} +1, & \text{if } \vec{w} \cdot \vec{x} + b > 0 \\ -1, & \text{else} \end{cases} \tag{3.4}$$

Each such linear threshold functions corresponds to a hyperplane in feature space. The sign function $sign\{\}$ returns 1 for a positive argument and -1 for a non-positive argument. This means that the side of the hyperplane on which an example $\vec{x}$ lies determines how it is classified by $h(\vec{x})$.

The conventional way of building a structure of increasing VC-dimension works by restricting the number of features. Linear threshold functions with N features have a VC-dimension of $N+1$ [Vapnik, 1998]. Given a ranked list of features, linear threshold functions using only the first feature have a VC-dimension of 2, those using only the first two features have a VC-dimension of 3, etc. However, this strategy fails if the learning task requires a large number of features. Furthermore, it is not clear how to define a ranked list of features a priori. Instead of restricting the number of features, support vector machines use a refined structure which does not necessarily depend on the dimensionality of the input space. Vapnik showed that margin, i.e. the distance δ from the hyperplane to the closest examples, can be used to upper bound the VC-dimension independent of the number of features. The exact definition of margin and its use in support vector machines is discussed in the following.

While independence of the number of features is an interesting property for text classification, it is only a superficial justification and does not yet guarantee good predictive performance. Intuitively, this property does not imply that SVMs will *necessarily* do well on high-dimensional learning tasks, but merely that they will not *necessarily* fail. A connection between SVMs and the properties of text classification that sufficiently implies good generalization gives Chapter 4.

3.1 Linear Hard-Margin SVMs

For simplicity, let us assume that the training data can be separated by at least one hyperplane h'. This means that there is a weight vector $\vec{w}'$ and a threshold b', so that all positive training examples are on one side of the hyperplane, while all negative training examples lie on the other side. This is equivalent to requiring

$$y_i \left[\vec{w}' \cdot \vec{x}_i + b'\right] > 0 \tag{3.5}$$

for each training example $(\vec{x}_i, y_i)$. In general, there can be multiple hyperplanes that separate the training data without error. This is depicted on the left-hand side of Figure 3.1. From these separating hyperplanes the support vector machine chooses the one with the largest margin δ. This particular hyperplane $h(\vec{x}^*)$ is shown in the right-hand picture of Figure 3.1. The margin δ is the distance from the hyperplane to the closest training examples. For each separable training set, there is only one hyperplane with maximum margin. The examples closest to the hyperplane are called support vectors. They have a distance of exactly δ. In Figure 3.1 the support vectors are marked with circles.

Finding the hyperplane with maximum margin can be translated into the following optimization problem:

Optimization Problem 1 (Hard-Margin SVM (primal))

$$\textit{minimize:} \qquad V(\vec{w}, b) = \frac{1}{2}\,\vec{w} \cdot \vec{w} \tag{3.6}$$

$$\textit{subject to:} \qquad \forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 \tag{3.7}$$

The constraints (3.7) formalize that all training examples should lie on the correct side of the hyperplane. Using the value of 1 on the right-hand side of the inequalities

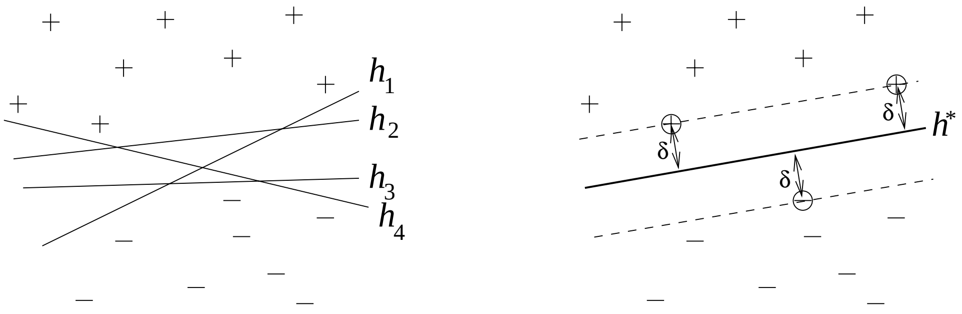


Figure 3.1: A binary classification problem in two dimensions. Positive examples are marked by $+$, negative examples by $-$. Left: many hyperplanes separate the training examples without error. Right: support vector machines find the hyperplane h^*, which separates the positive and negative training examples with maximum margin δ. The examples closest to the hyperplane are called *support vectors* (marked with circles).

instead of 0 enforces a certain distance δ (i.e. margin) from the hyperplane. It is easy to verify that

$$\delta = \frac{1}{||\vec{w}||} \tag{3.8}$$

$||\vec{w}||$ denotes the L_2-norm of $\vec{w}$. Therefore, minimizing $\vec{w} \cdot \vec{w}$ is equivalent to maximizing margin. The weight vector $\vec{w}$ and the threshold b solving Optimization Problem 1 describe the maximum-margin hyperplane.

How do support vector machines implement structural risk minimization? Vapnik showed that there is a connection between the margin and the VC-dimension.

Lemma 1 ([Vapnik, 1982] VC-dimension of Margin Hyperplanes) *Consider hyperplanes $h(\vec{x}) = sign\{\vec{w} \cdot \vec{x} + b\}$ in an N dimensional space as hypotheses. If all example vectors x_i are contained in a ball of radius R and it is required that for all examples x_i*

$$abs(\vec{w} \cdot \vec{x}_i + b) \geq 1 \tag{3.9}$$

then this set of hyperplanes has a VC-dimension d bounded by

$$d \leq min(\left[\frac{R^2}{\vec{w}^2}\right], N) + 1 \tag{3.10}$$

The lemma states that the VC-dimension is lower the larger the margin. Note that the VC-dimension of maximum-margin hyperplanes does not necessarily depend on the number of features! Instead the VC-dimension depends on the Euclidean length $||\vec{w}||$ of the weight vector $\vec{w}$ optimized by the support vector machine. Intuitively, this means that the true error of a separating maximum-margin hyperplane is close to the training error even in high-dimensional spaces, if it has a small weight vector. However, bound (3.2) does not directly apply to support vector machines, since the VC-dimension depends on the location of the examples [Vapnik, 1995]. The bounds in [Shawe-Taylor et al., 1996] account for this data dependency. An overview is given in [Cristianini and Shawe-Taylor, 2000]. A different justification for the generalization performance of support vector machines in terms of the expected generalization error $\mathcal{E}(Err(h_{SVM})$ is discussed in Chapter 4.

Numerically, Optimization Problem 1 can be difficult to handle. Therefore, the following Wolfe-dual [Fletcher, 1987] quadratic optimization problem is commonly solved in practice [Vapnik, 1998]. It has the same solution as Optimization Problem 1.

Optimization Problem 2 (Hard-Margin SVM (dual))

$$\textit{minimize:} \quad W(\vec{\alpha}) = -\sum_{i=1}^{n} \alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j) \tag{3.11}$$

$$\textit{subject to:} \quad \sum_{i=1}^{n} y_i \alpha_i = 0 \tag{3.12}$$

$$\forall i \in [1..n] : 0 \leq \alpha_i \tag{3.13}$$

The matrix Q with $Q_{ij} = y_i y_j (\vec{x}_i \cdot \vec{x}_j)$ is commonly referred to as the Hessian. An efficient algorithm for solving this type of optimization problem is developed in Chapter 8. With this algorithm it is possible to train SVMs even on large data sets containing several ten-thousands of examples and attributes. The result of the optimization process is a vector of coefficients $\vec{\alpha}^T = (\alpha_1, \ldots, \alpha_n)^T$ for which (3.11) is minimum. These coefficients can be used to construct the hyperplane solving Optimization Problem 1.

$$\vec{w} \cdot \vec{x} = \left(\sum_{i=1}^{n} \alpha_i y_i \vec{x}_i\right) \cdot \vec{x} = \sum_{i=1}^{n} \alpha_i y_i (\vec{x}_i \cdot \vec{x}) \quad \text{and} \quad b = y_{sv} - \vec{w} \cdot \vec{x}_{sv} \tag{3.14}$$

Equation (3.14) shows that the resulting weight vector of the hyperplane is constructed as a linear combination of the training examples. Only support vectors have a coefficient α_i that is non-zero. To calculate b from the solution of Optimization Problem 2, an arbitrary support vector $\vec{x}_{sv}$ with its class label y_{sv} can be used.

A variant of the general SVM formulation is the following. Sometimes it is required for simplicity reasons that the hyperplane passes through the origin of the coordinate system. Such hyperplanes will be called *unbiased hyperplanes* in the following. If it is necessary to emphasize the difference, general hyperplanes not necessarily passing through the origin will be called *biased hyperplanes*. A hyperplane passing through the origin can be enforced by using $b = 0$ in Optimization Problem 1. The corresponding dual is the same as Optimization Problem 2 without the equality constraint (3.12).

3.2 Soft-Margin SVMs

One problem with the simple formulation above is that training fails when the training examples are not linearly separable. A solution to Optimization Problems 1 and 2 does not exist in this case. Even though most text-classification problems are linearly separable, it might still be preferable to allow some errors on the training data, as indicated by structural risk minimization. Cortes and Vapnik suggest a solution to this problem [Cortes and Vapnik, 1995]. It is called the *soft-margin* SVM. They include an upper bound on the number of training errors in the objective function of Optimization Problem 1. Then they minimize this upper bound and the length of the weight vector simultanously.

Optimization Problem 3 (Soft-Margin SVM (primal))

$$\textit{minimize:} \quad V(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\, \vec{w} \cdot \vec{w} + C \sum_{i=1}^{n} \xi_i \tag{3.15}$$

$$\textit{subject to:} \quad \forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i \tag{3.16}$$

$$\forall_{i=1}^{n} : \xi_i > 0 \tag{3.17}$$

The ξ_i are called slack variables. If a training example lies on the "wrong" side of the hyperplane, the corresponding ξ_i is greater than 1. Therefore $\sum_{i=1}^{n} \xi_i$ is an upper bound on the number of training errors. The factor C in (3.15) is a parameter that allows one to trade off training error vs. model complexity. A small value for C will increase the number of training errors, while a large C will lead to a behaviour similar to that of a hard-margin SVM.

For computational reasons it is again useful to solve the Wolfe dual of Optimation Problem 3 instead of solving Optimization Problem 3 directly. It is of the same form as Optimization Problem 2. The only difference is that C upper-bounds the values of the α_i.

Optimization Problem 4 (Soft-Margin SVM (dual))

$$\textit{minimize:} \qquad W(\vec{\alpha}) = -\sum_{i=1}^{n} \alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j) \tag{3.18}$$

$$\textit{subject to:} \qquad \sum_{i=1}^{n} y_i \alpha_i = 0 \tag{3.19}$$

$$\forall i \in [1..n] : 0 \leq \alpha_i \leq C \tag{3.20}$$

Again, all training examples with $\alpha_i > 0$ are called support vectors. To differentiate between those with $0 < \alpha_i < C$ and those with $\alpha_i = C$, the former will be called *unbounded support vectors* while the latter will be called *bounded support vectors*. From the solution of Optimization Problem 4 the classification rule can be computed as

$$\vec{w}\cdot\vec{x} = \left(\sum_{i=1}^{n} \alpha_i y_i \vec{x}_i\right) \cdot \vec{x} = \sum_{i=1}^{n} \alpha_i y_i (\vec{x}_i \cdot \vec{x}) \quad \text{and} \quad b = y_{usv} - \vec{w}\cdot\vec{x}_{usv} \tag{3.21}$$

like in the hard-margin case. The only additional restiction is that the support vector $(\vec{x}_{usv}, y_{usv})$ for calculating b has to be an unbounded support vector. While it is highly unlikely in practice that one gets a solution of Optimization Problem 4 with only bounded support vectors, it is theoretically possible (see [Burges and Crisp, 1999] for a discussion). In this case the solution of the SVM will be called *unstable*, since the hyperplane is not uniquely determined. In particular, b can take any value in a certain interval. If there is at least one unbounded support vector, the solution is called *stable*. This definition of stability is used in Chapter 5.

By reformulating the training problem, it is possible to ensure that the solution of a SVM is stable.

Optimization Problem 5 (Stablized Soft-Margin SVM (primal))

$$\textit{minimize:} \qquad V(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n} \xi_i \tag{3.22}$$

$$\textit{subject to:} \qquad \forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i \tag{3.23}$$

$$[\vec{w}\cdot\vec{x}'_1 + b] \geq 1 \tag{3.24}$$

$$-[\vec{w}\cdot\vec{x}'_2 + b] \geq 1 \tag{3.25}$$

$$\forall_{i=1}^{n} : \xi_i > 0 \tag{3.26}$$

The $\vec{x}'_1$ and $\vec{x}'_2$ can be viewed as artificial training examples. They are orthogonal to all training examples and also $\vec{x}'_1 \cdot \vec{x}'_2 = 0$. Their length is $||\vec{x}'_1|| = ||\vec{x}'_2|| = \kappa$. $\kappa > 0$ is a parameter. It is easy to verify that at least one of $\vec{x}'_1$ and $\vec{x}'_2$ must be an unbounded support vector. The Wolfe dual is:

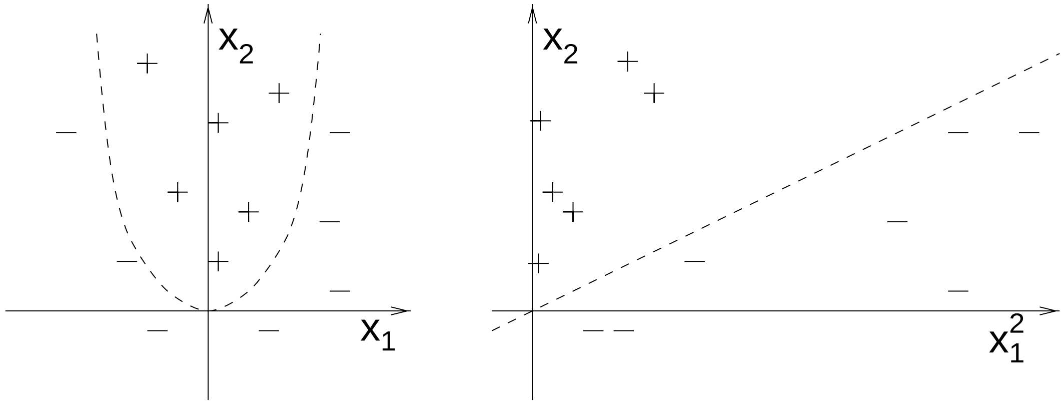


Figure 3.2: The left-hand graph shows a training sets that is not linearly separable in (x_1, x_2). The right-hand graph depicts the same problem after a non-linear transformation, now projected onto (x_1^2, x_2). In this new space, the training examples are linearly separable.

Optimization Problem 6 (Stablized Soft-Margin SVM (dual))

$$\textit{maximize:}\quad W(\vec{\alpha},\vec{\beta}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \mathcal{K}(\vec{x}_i,\vec{x}_j) + \beta_1 + \beta_2 - \frac{1}{2}(\beta_1^2+\beta_2^2)\kappa^2 \quad (3.27)$$

$$\textit{subject to:}\quad \beta_1 - \beta_2 + \sum_{i=1}^{n} y_i\alpha_i = 0 \quad (3.28)$$

$$\forall i : 0 \leq \alpha_i \leq C \quad (3.29)$$

$$0 \leq \beta_1 \wedge 0 \leq \beta_2 \quad (3.30)$$

3.3 Non-Linear SVMs

So far in this presentation SVMs were discussed only for linear classification rules. Linear classifiers are inappropriate for many real-world problems, since they have a non-linear structure. A remarkable property of SVMs is that they can easily be transformed into non-linear learners [Boser et al., 1992]. In principle, the approach used is as follows. The attribute vectors $\vec{x}_i$ are mapped into a high-dimensional feature space X' using a non-linear mapping $\Phi(\vec{x}_i)$. The SVM then learns the maximum-margin linear classification rule in feature space X'. Despite the fact that the classification rule is linear in X', it is non-linear when projected into the original input space.

The following example with two input attributes x_1 and x_2 illustrates this. Let us choose

$$\Phi((x_1,x_2)^T) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)^T \quad (3.31)$$

as the non-linear mapping. Although it is not possible to linearly separate the examples in the left part of Figure 3.2, they are separable with a linear function after mapping them into the feature space using $\Phi(\vec{x})$ (right part of Figure 3.2). One linear separator (although not the maximum-margin separator) is the weight vector $(-1, 0, 0, 0, \sqrt{2}, 0)^T$ with $b = 0$ as displayed in the both parts of Figure 3.2.

In general such a mapping $\Phi(\vec{x})$ is inefficient to compute. Boser et al. [Boser et al., 1992] have discovered a special property of SVMs that solves this problem. Both during training and testing, it is sufficient to be able to compute dot-products in the feature space,

i. e. $\Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)$. For special mappings $\Phi(\vec{x})$ such dot-products can be computed very efficiently using *kernel functions* $\mathcal{K}(\vec{x}_1, \vec{x}_2)$. If a function $\mathcal{K}(\vec{x}_1, \vec{x}_2)$ satisfies the condition of Mercer's Theorem (see [Vapnik, 1995]), it is guaranteed to compute the inner product of the vectors $\vec{x}_1$ and $\vec{x}_2$ after they have been mapped into a new "feature" space by some non-linear mapping Φ:

$$\Phi(\vec{x}_1) \cdot \Phi(\vec{x}_2) = \mathcal{K}(\vec{x}_1, \vec{x}_2) \tag{3.32}$$

Depending on the choice of kernel function, SVMs learn polynomial classifiers, radial basis function (RBF) classifiers, or two layer sigmoid neural nets.

$$K_{poly}(\vec{x}_1, \vec{x}_2) = (\vec{x}_1 \cdot \vec{x}_2 + 1)^d \tag{3.33}$$

$$K_{rbf}(\vec{x}_1, \vec{x}_2) = exp(-\gamma(\vec{x}_1 - \vec{x}_2)^2) \tag{3.34}$$

$$K_{sigmoid}(\vec{x}_1, \vec{x}_2) = tanh(s(\vec{x}_1 \cdot \vec{x}_2) + c) \tag{3.35}$$

The kernel for the example above is $K_{poly}(\vec{x}_1, \vec{x}_2) = (\vec{x}_1 \cdot \vec{x}_2 + 1)^2$. Clearly, computing the kernel function is much more efficient than, for example, enumerating all polynomial terms like in polynomial regression. To use a kernel function, one simply substitutes every occurrence of the inner product in equations (3.18) and (3.21) with the desired kernel function.

3.4 Asymmetric Misclassification Cost

In text classification, the number of negative examples is often much larger than the number of positive examples. In this situation, consider a simple default classifier that always predicts "negative". As noted earlier, in terms of error rate this default classifier performs excellently and is difficult to beat. But, clearly, such a classifier is of little use. In practice, one would like to penalize errors on positive examples stronger than errors on negative examples. This can be achieved using cost factors C_{-+} and C_{+-} to adjust the cost of false positives vs. false negatives. Such cost factors can be directly incorporated into the SVM. Finding the hyperplane that minimizes empirical cost can be translated into the following Optimization Problem [Morik et al., 1999]:

Optimization Problem 7 (Soft-Margin SVM with Asymmetric Cost (primal))

$$\text{minimize:} \quad V(\vec{w}, b, \vec{\xi}) = \frac{1}{2}||\vec{w}||^2 + C_{-+}\sum_{i:y_i=1} \xi_i + C_{+-}\sum_{j:y_j=-1} \xi_j \tag{3.36}$$

$$\text{subject to:} \quad \forall k : y_k[\vec{w} \cdot \vec{x}_k + b] \geq 1 - \xi_k \tag{3.37}$$

The dual problem is equivalent to Optimization Problem 4 after replacing the upper bounds on α_i with C_{-+} and C_{+-} for positive and negative examples respectively.

3.5 Other Maximum-Margin Methods

While this dissertation mainly discusses support vector machines, other learning methods also optimize margin. For example, the success of boosting algorithms like Adaboost [Freund and Schapire, 1996] can be explained in terms of margins as well [Schapire et al., 1997]. Unlike SVMs, boosting does not consider the L_2-norm of the weight vector, but is connected to its L_1-norm [Grove and Schuurmans, 1998].

It is straightforward to generalize Optimization Problem 3 by replacing the L_2-norm with another norm.

Optimization Problem 8 (Generic Maximum-Margin Method)

$$\textit{minimize:} \quad V(\vec{w}, b, \vec{\xi}) = \|\vec{w}\|_l + C \sum_{i=1}^{n} \xi_i \tag{3.38}$$

$$\textit{subject to:} \quad \forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i \tag{3.39}$$

$$\forall_{i=1}^{n} : \xi_i > 0 \tag{3.40}$$

An obvious choice is $l = 1$ like in [Bennett and Demiriz, 1998]. L_1 should give preference to sparser weight vectors. This should be most appropriate when the learning task has a few strong features, as previously demonstrated for the Winnow algorithm [Kivinen et al., 1997].

This dissertation mainly discusses support vector machines. However, it is likely that, depending on the properties of the particular text classification task, other maximum-margin methods (e.g. using a different norm) can be equally or more suitable.

3.6 Further Work

Support vector machines and other maximum-margin methods like boosting are a currently very active area of research. This chapter introduces only those results providing the basis for the work in this dissertation. Each following chapter will give additional information and related work on their relevant aspects. In particular, chapters 4 and 5 discuss learning theory, while chapters 8 and 9 deal with algorithms for training SVMs.

A more detailed introduction to SVMs is Chris Burges's tutorial [Burges, 1998]. A more general overview of the state-of-the-art can be found in the recent book [Cristianini and Shawe-Taylor, 2000] on support vector machines.

Part I

Theory

Chapter 4

A Statistical Learning Model of Text Classification

There are at least two ways to motivate why a particular learning method is suitable for a particular learning task. Since ultimately one is interested in the performance of the method, one way to motivate a new method is through comparative studies. Chapters 6 and 7 present such studies and show that SVMs deliver state-of-the-art classification performance. However, success on benchmarks is a brittle justification for a learning algorithm and gives only limited insight. Therefore, this dissertation takes a different approach. It introduces support vector machines for learning text classifiers from a theoretical perspective.

This chapter develops a theoretical learning model of text classification. For the first time, it is possible to connect the statistical properties of text-classification tasks with the generalization performance of a learner — namely the SVM. Unlike conventional approaches to learning text classifiers, which rely primarily on empirical evidence, this model explains why and when SVMs perform well for text classification. In particular, it addresses the following questions: Why can support vector machines handle the large feature spaces in text classification effectively? How is this related to the statistical properties of text? What are sufficient conditions for applying SVMs to text-classification problems successfully?

Answering these questions is the subject of this chapter. It presents an abstract model of text-classification tasks. This model is based on statistical properties of text-classification problems that are both observable and intuitive, combining results from information retrieval, computational linguistics, and statistical learning theory. Using this model, it is possible to prove what types of text-classification problems are efficiently learnable with SVMs. In particular, the result is an upper bound connecting the expected generalization error of a support vector machine with the statistical properties of a particular text-classification task.

This chapter is structured as follows. The following section will identify the key properties of text-classification tasks. They motivate the model formally defined in Section 4.2. In addition to verifying the assumptions of the model against real text-classification tasks, this section proves the learnability results. Section 4.3 further validates the model against experimental data before Section 4.4 analyzes the complexity of text-classification tasks and identifies sufficient conditions for goods generalization performance.

4.1 Properties of Text-Classification Tasks

To make useful statements about why a particular learning methods should work well for text classification, it is necessary to identify key properties of text-classification tasks. These properties should hold over a large range of text-classification tasks, should be intuitive and observable for a given collection of documents, and should imply good learnability using an appropriate measure for success. The following properties are meant to motivate the model that is developed in Section 4.2.2. Since they mainly serve as the motivation, their description and verification is kept on an intuitive level. Their formalization and stringent verification is part of building and validating the model in sections 4.2 and 4.3.

4.1.1 High-Dimensional Feature Space

Independent of the particular choice of terms, text-classification problems involve high-dimensional feature spaces. If each word occurring in the training documents is used as a feature, text-classification problems with a few thousand training examples can lead to 10,000 and more dimensions. Consider the three document collections described in Section 2.7.1. For the Reuters data, 27,658 distinct words occur within the 9,603 training documents at least once. Similarly, the 3,957 WebKB documents lead to 38,359 features and the 10,000 Ohsumed documents contain 38,679 distinct words. The dimensionality can be reduced using stopword removal and stemming. Nevertheless, the resulting feature set will still have a size of the same order of magnitude.

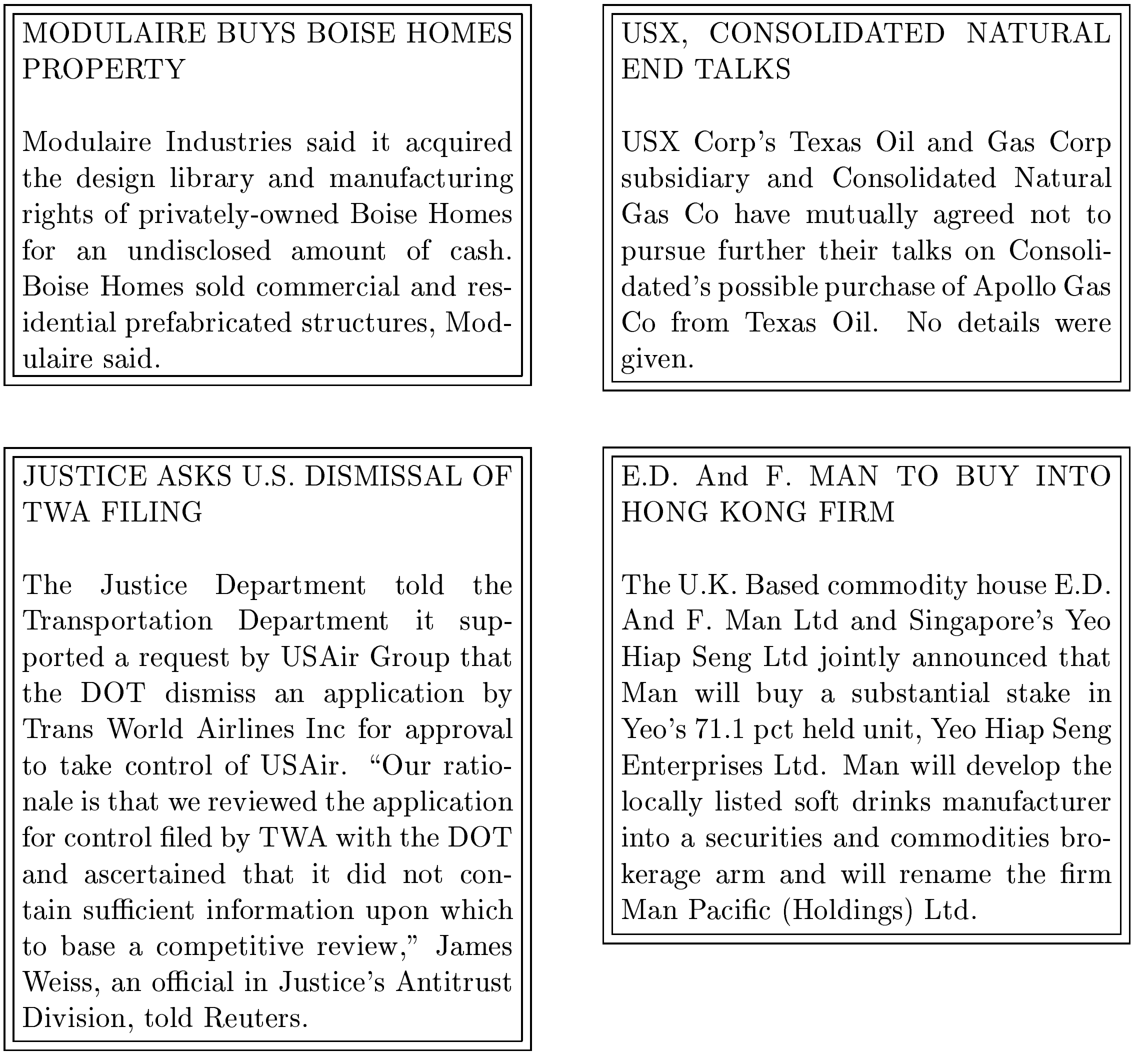
These feature-set sizes are not exceptional artifacts of the particular test collections considered here. Independent of the type of text, there is a stable connection between the size of a document and the number of distinct words that occur in it. It is commonly called Heaps's law [Heaps, 1978]. It states that the number of distinct words V is related to the total number of words in the documents s by

$$V = k\, s^{\beta}, \tag{4.1}$$

where k and β depend on the particular text and s is sufficiently large. Typically, k is between 10 and 100 [Baeza-Yates and Ribeiro-Neto, 1999]. Common values for β are between 0.4 and 0.6 [Araújo et al., 1997, Baeza-Yates and Navarro, 1997]. Treating a collection of documents as its concatenation into one large body of text makes it possible to analyze the feature-set size for text classification. For $k = 15$ and $\beta = 0.5$, it is easy to calculate that the feature-set size for a collection of 10,000 documents having an average length of 50 words is approximately 35,000. This accurately reflects the experimental findings.

4.1.2 Heterogenous Use of Terms

One way to avoid these high-dimensional input spaces is to assume that only a few features are relevant and necessary for the task. Feature selection tries to exclude all irrelevant features. However, in text categorization this can easily lead to a loss of information, since there are often many relevant features. Consider the 4 documents shown in Figure 4.1. All documents are Reuters articles from the category "corporate acquisitions". Nevertheless, the overlap between their document vectors is very small. In this extreme case, the documents do not share any content words. The only words that occur in at least two documents are "*it*", "*the*", "*and*", "*of*", "*for*", "*an*", "*a*", "*not*", "*that*", and "*in*". All these words are stopwords and it is unlikely that they help discriminate between documents



MODULAIRE BUYS BOISE HOMES PROPERTY

Modulaire Industries said it acquired the design library and manufacturing rights of privately-owned Boise Homes for an undisclosed amount of cash. Boise Homes sold commercial and residential prefabricated structures, Modulaire said.

USX, CONSOLIDATED NATURAL END TALKS

USX Corp's Texas Oil and Gas Corp subsidiary and Consolidated Natural Gas Co have mutually agreed not to pursue further their talks on Consolidated's possible purchase of Apollo Gas Co from Texas Oil. No details were given.

JUSTICE ASKS U.S. DISMISSAL OF TWA FILING

The Justice Department told the Transportation Department it supported a request by USAir Group that the DOT dismiss an application by Trans World Airlines Inc for approval to take control of USAir. "Our rationale is that we reviewed the application for control filed by TWA with the DOT and ascertained that it did not contain sufficient information upon which to base a competitive review," James Weiss, an official in Justice's Antitrust Division, told Reuters.

E.D. And F. MAN TO BUY INTO HONG KONG FIRM

The U.K. Based commodity house E.D. And F. Man Ltd and Singapore's Yeo Hiap Seng Ltd jointly announced that Man will buy a substantial stake in Yeo's 71.1 pct held unit, Yeo Hiap Seng Enterprises Ltd. Man will develop the locally listed soft drinks manufacturer into a securities and commodities brokerage arm and will rename the firm Man Pacific (Holdings) Ltd.

Figure 4.1: Four documents from the Reuters "corporate acquisitions" category that do not share any content words.

about corporate acquisitions and other documents. This implies that it is necessary to consider at least four other words as features to at least somewhat describe the content of each document.

The sample of documents in Figure 4.1 was selected to be rather extreme. However, even a random sample of documents is unlikely to contain a single feature discriminating positive from negative examples. There is generally not a small set of words or even a single word that sufficiently describes all documents with respect to the classification task. However, a weaker relationship holds that can be described by what Wittgenstein calls "family resemblance" [Wittgenstein, 1967, aphorism 67]. Imagine three brothers Al, Jack, and Elmer. Al and Jack have the same eyebrows and the same hair, while Elmer and Bob have the same ears, and Al and Bob have the same smile and nose. There is no common feature among them yet they all resemble each other. Similarly, related documents share keywords, but there is no term common to all related documents. Documents from the same category can consist of different words, since natural language allows the expression of related content with different formulations. In the most extreme case, two words can

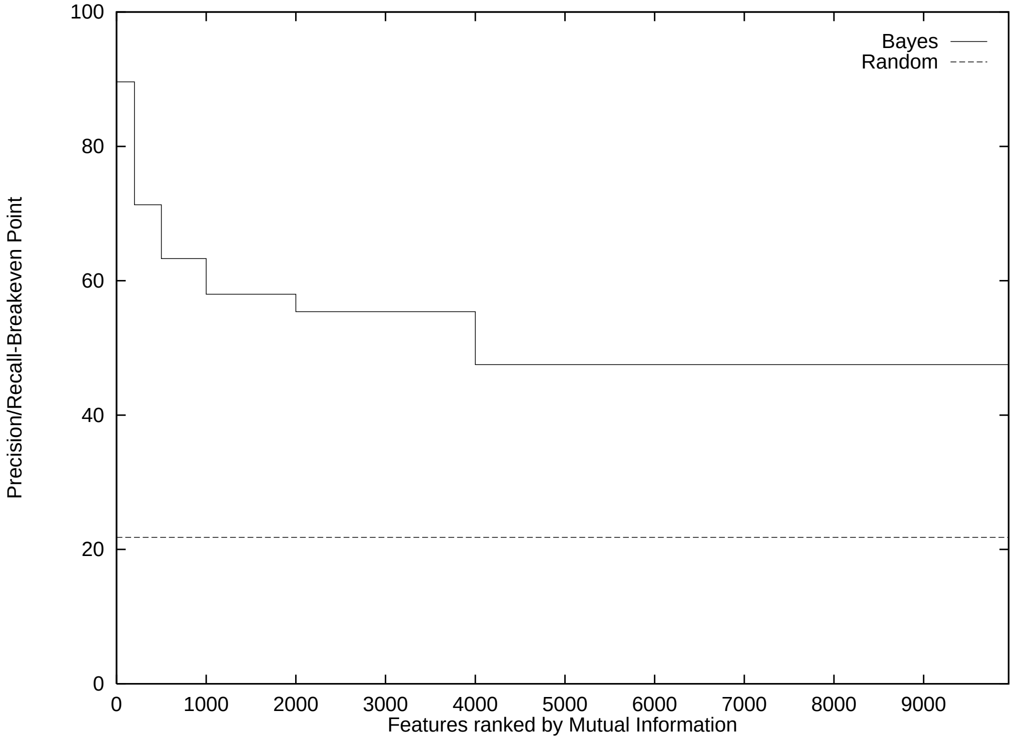


Figure 4.2: Learning without using the "best" features.

be synonyms – different words with the same or similar meanings – that can substitute for one another (e.g. "buy", "acquire", "purchase" ...). It is therefore necessary to consider a fairly large set of words. Each word indicates to some extent which category a document belongs to. In the example from Figure 4.1 such words are for example "*acquired*", "*purchase*", "*antitrust*", "*buy*", and "*stake*", but also "*undisclosed*", "*amount*", "*cash*", "*justice*", "*department*", and "*rename*". This is a motivation for considering a fairly large set of features to sufficiently describe the content of all documents.

4.1.3 High Level of Redundancy

While there are generally many different features relevant to the classification task, often several such cues occur in one document. These cues are partly redundant. Figure 4.2 shows the results of an experiment on the Reuters "corporate acquisitions" category. All features (after stemming and stopword removal) are ranked according to their (binary) empirical mutual information with the class label (cf. Section 2.3.1). The higher the mutual information of a word, the better the simple classifier that classifies according to whether that particular word occurs in a document perform. Then a naive Bayes classifier is trained using only those features ranked 1-200, 201-500, 501-1000, 1001-2000, 2001-4000, 4001-9947. The results in Figure 4.2 show that even features ranked lowest still contain considerable information and are somewhat relevant. A classifier using only those "worst" features has a performance much better than random.

This behavior can be explained as follows. Most documents contain more than one word indicating its class. Even after removing the best features, the remaining words still describe the content of many documents to some extent. This means that document vectors are redundant with respect to the classification task. This is also supported by the success of distributional word clustering [Baker and McCallum, 1998]. Many words have a similar distribution with respect to the learning task and can be treated like synonyms.

4.1.4 Sparse Document Vectors

While there is a large space of potential features, each document contains only a small number of distinct words. The Reuters documents are on average 152 words long and contain 74 distinct words. Similarly, WebKB documents are on average 277 words long and contain 130 distinct words, and Ohsumed articles are 209 words long and contain 100 distinct words. This implies that document vectors are very sparse. Only a few words occur with non-zero frequency.

4.1.5 Frequency Distribution of Words and Zipf's Law

The occurrence frequencies of words in natural-language text behave in a very stable way. A first approximation modeling the distribution of term frequencies is Zipf's law [Zipf, 1949]. Zipf's law states that if one ranks words by their term frequency, the r-th most frequent words occurs $\frac{1}{r}$ times the term frequency of the most frequent words. This implies that there is a small number of words that occurs very frequently, while most words occur very infrequently. For example, with a vocabulary of 10,000 words, the 100 most frequent words account for approximately 50 percent of all occurrences. While Zipf's law does capture the basic behavior of term frequencies based on their rank, experimental data [Araújo et al., 1997] suggests that Mandelbrot distributions [Mandelbrot, 1959, Miller et al., 1958]

$$TF_i = \frac{c}{(k+r)^{\phi}} \tag{4.2}$$

with parameters c, k, and ϕ provide a better fit. This kind of connection between frequency rank r and term frequency TF_r will be called (generalized) Zipf's law in what follows. Figure 4.3 plots term frequency vs. frequency rank for the Reuters, the WebKB, and the Ohsumed collection. The dashed lines are approximations $TF_r = \frac{800000}{(r+5)^{1.3}}$ for Reuters, $TF_r = \frac{470000}{(r+5)^{1.25}}$ for WebKB, and $TF_r = \frac{1100000}{(r+5)^{1.3}}$ for the Ohsumed collection. The approximations model the true behavior of the word frequencies accurately. Note that besides the constant factor reflecting the size of the document collection, the approximations are very similar across all three collections. Furthermore, if one assumes that the document collection is homogeneous, it is easy to verify that every sufficiently large subset of the collection will also follow the same generalized Zipf's law with the constant factor scaled according to the size of the subset. So it is reasonable to assume that the frequency distribution of words in each individual document also follows the respective Zipf's law approximately.

4.1.6 Linear Separability and Large Margin

All Ohsumed categories, all WebKB tasks, and most Reuters categories are linearly separable. This means that there is a hyperplane so that all positive examples are on one side of the hyperplane, while all negative examples are on the other. Inseparability on some Reuters categories is often due to dubious documents (consisting only of a headline) or obvious misclassifications of the human indexers.

Even more important is the observation that separability is possible with a large margin. Table 4.1 shows the size of the normalized inverse margin $\frac{R^2}{\delta^2}$ for the ten most frequent Reuters categories, the WebKB categories, and the five most frequent Ohsumed

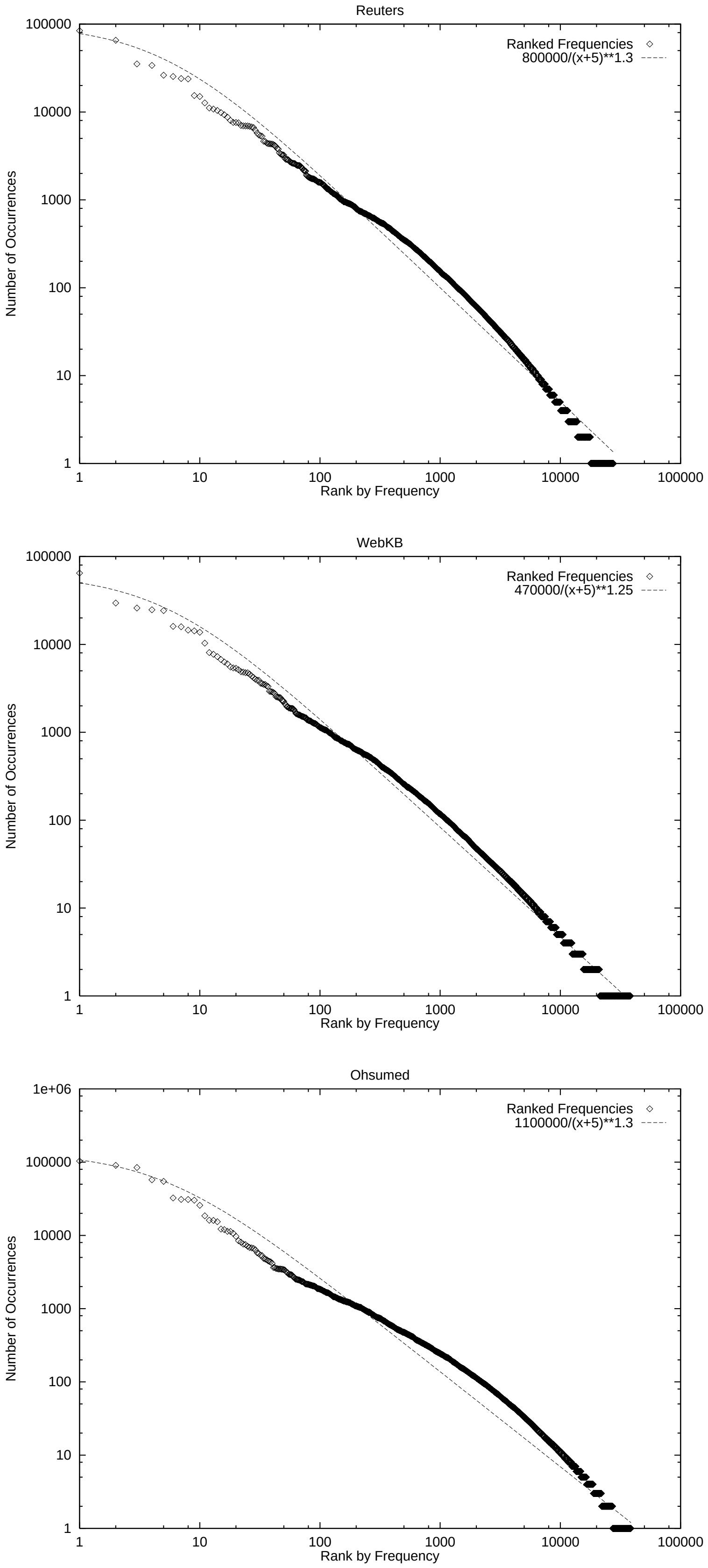


Figure 4.3: Distribution of term frequencies in the Reuters, WebKB, and Ohsumed collection. The dashed line is an approximation of the observed curve using a Mandelbrot distribution.

Reuters	$\frac{R^2}{\delta^2}$	$\sum_{i=1}^{n} \xi_i$
earn	1143	0
acq	1848	0
money-fx	1489	27
grain	585	0
crude	810	4
trade	869	9
interest	2082	33
ship	458	0
wheat	405	2
corn	378	0

WebKB	$\frac{R^2}{\delta^2}$	$\sum_{i=1}^{n} \xi_i$
course	519	0
faculty	1636	0
project	741	0
student	1588	0

Ohsumed	$\frac{R^2}{\delta^2}$	$\sum_{i=1}^{n} \xi_i$
Pathology	11614	0
Cardiovascular	4387	0
Neoplasms	2868	0
Nervous System	3303	0
Immunologic	2556	0

Table 4.1: Normalized inverse margin and training loss for the Reuters, the WebKB, and the Ohsumed data for $C = 50$. As determined by the model-selection experiments in Chapter 6, TFIDF-weighting is used for Reuters and Ohsumed, while the representation for WebKB is binary. No stemming is performed and stopword removal is used only on the Ohsumed data.

categories. R is the radius of the ball containing the training examples. Since all document vectors are normalized to unit length, R is bounded by 1. Note that a small normalized inverse margin $\frac{R^2}{\delta^2}$ corresponds to a large margin δ. The tables also include the training loss $\sum_{i=1}^{n} \xi_i$. For those categories that are not separable, the training loss is greater than zero. Nevertheless, the training loss is small, indicating that only a few documents hinder linear separation. The training loss is an upper bound for the number of documents that need to be removed so that the data is separable.

The following section will show that a large expected margin and a small expected training loss are sufficient conditions for learnability. These two quantities will be the proxies for connecting the other properties of text-classification tasks with the generalization performance of the support vector machine.

4.2 A Discriminative Model of Text Classification

The goal of this section is a statistical learning model of text-classification tasks. Using a two step approach, it provides the relationship between the properties of text-classification tasks identified above and the expected error rate of an SVM.

The first step shows that large margin combined with low training error is a sufficient condition for good generalization accuracy. While large margin and low training error can be observed over a variety of different text-classification tasks, this property is not observable or accessible before training is performed. Furthermore, the large margin property is less intuitive than the other properties identified in the previous section. Therefore, the second step shows how the other properties of text-classification tasks – namely sparse document vectors, Zipf's law, and redundancy in a large relevant feature space – necessarily lead to large-margin separation. The overall structure of the argument is illustrated in Figure 4.4.

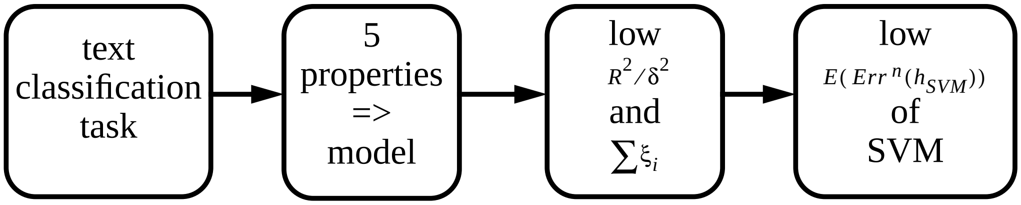

text classification task
5 properties => model
low R^2/δ^2 and $\sum\xi_i$
low $E(Err^n(h_{SVM}))$ of SVM

Figure 4.4: Structure of the argument.

4.2.1 Step 1: Bounding the Expected Error Based on the Margin

This section shows that large margin combined with low training error leads to high generalization accuracy. It uses the bound on the number of leave-one-out errors from Chapter 5. This bound can not only be used to design efficient estimators, but also leads to a bound on the expected error.

Vapnik has already presented two bounds on the expected error [Vapnik, 1998, page 414]. His first bound connects the expected error with the expected margin. But this bound is restricted in two ways: it applies only to unbiased hyperplanes (i. e. hyperplanes passing through the origin), and it assumes that the training data is always perfectly separable. Both restrictions are removed by the bound below. It applies to all stable soft-margin SVMs. Vapnik's second bound connects the expected error with the expected number of support vectors. While that bound applies to biased optimal hyperplanes, it is less tight than the bound below.

Theorem 1 (Bound on Expected Error of Soft-Margin SVMs) *The expected error rate $\mathcal{E}(Err^n(h_{SVM}))$ of a soft-margin SVM based on n training examples with $c \leq \mathcal{K}(\vec{x}_i, \vec{x}_j) \leq c + R^2$ for some constant c, is bounded by*

$$C \geq \frac{1}{\rho\, R^2} \quad : \quad \mathcal{E}(Err^n(h_{SVM})) \leq \frac{\rho\, \mathcal{E}\Big(\frac{R^2}{\delta^2}\Big) + C\,\rho\, R^2 \mathcal{E}\Big(\sum_{i=1}^{n+1} \xi_i\Big)}{n+1} \tag{4.3}$$

$$C < \frac{1}{\rho\, R^2} \quad : \quad \mathcal{E}(Err^n(h_{SVM})) \leq \frac{\rho\, \mathcal{E}\Big(\frac{R^2}{\delta^2}\Big) + \rho\,(CR^2+1)\mathcal{E}\Big(\sum_{i=1}^{n+1} \xi_i\Big)}{n+1} \tag{4.4}$$

For unbiased hyperplanes ρ equals 1, and for stable hyperplanes ρ equals 2. The expectations on the right are over training sets of size $n+1$.

Proof *Lemma 4 establishes a bound on the leave-one-out error of soft-margin SVMs.*

$$Err_{loo}^{n+1}(S) \leq \frac{1}{n+1} |\{i : (\rho\, \alpha_i R^2 + \xi_i) \geq 1\}| \tag{4.5}$$

Depending on the value of C it is possible to relate this bound to a sum of the soft margin and the training loss.

Case $C \geq \frac{1}{\rho\, R^2}$: *Since ξ_i can only be non-zero if the corresponding $\alpha_i = C$, the upper bound reduces to*

$$\begin{aligned} Err_{loo}^{n+1}(S) &\leq \frac{1}{n+1} |\{i : (\rho\, \alpha_i R^2) \geq 1\}| && (4.6) \\ &\leq \frac{1}{n+1} \rho\, R^2 \sum_{i=1}^{n+1} \alpha_i && (4.7) \end{aligned}$$

At the solution of the soft-margin optimization problem, the value of the primal objective (3.15) and the dual objective (3.18) are equal.

$$\frac{1}{2}\vec{w}\cdot\vec{w}+C\sum_{i=1}^{n+1}\xi_i=\sum_{i=1}^{n+1}\alpha_i-\frac{1}{2}\vec{w}\cdot\vec{w} \tag{4.8}$$

Rearranging the previous equation and using $\vec{w}\cdot\vec{w}=\frac{1}{\delta^2}$ shows that the leave-out-out error is bounded by

$$Err_{loo}^{n+1}(S) \quad\leq\quad \frac{1}{n+1}\left[\rho\,\frac{R^2}{\delta^2}+C\,\rho\,R^2\sum_{i=1}^{n+1}\xi_i\right] \tag{4.9}$$

The application of Theorem 5 about the bias of the leave-one-out estimate completes the proof for this case.

Case $C<\frac{1}{\rho\,R^2}$: *Similarly, there is also a bound on the leave-one-out error in case $C<\frac{1}{\rho\,R^2}$, although it is looser.*

$$\begin{aligned} Err_{loo}^{n+1}(S) &\leq \frac{1}{n+1}|\{i:(\rho\,\alpha_i R^2+\xi_i)\geq 1\}| && (4.10)\\ &\leq \frac{1}{n+1}\left[\rho\,R^2\sum_{i=1}^{n+1}\alpha_i+\sum_{i=1}^{n+1}\xi_i\right] && (4.11)\end{aligned}$$

Using again the equality of the primal and the dual objective at the solution, it is easy to see that

$$\vec{w}\cdot\vec{w}+(C+\frac{1}{\rho R^2})\sum_{i=1}^{n+1}\xi_i=\sum_{i=1}^{n+1}\alpha_i+\frac{1}{\rho R^2}\sum_{i=1}^{n+1}\xi_i \tag{4.12}$$

Multiplying with $\rho\,R^2$, we can upper bound the leave-one-out error with

$$Err_{loo}^{n+1}(S) \quad\leq\quad \frac{1}{n+1}\left[\rho\,\frac{R^2}{\delta^2}+\rho\,(CR^2+1)\sum_{i=1}^{n+1}\xi_i\right] \tag{4.13}$$

Again, the application of Theorem 5 completes the proof. ▮

This bound is the first step towards learnability results for text classification with SVMs. It shows that the key quantities are the margin δ, the length of the document vectors R, and the training loss ξ.

4.2.2 Step 2: TCat-Concepts as a Model of Text-Classification Tasks

Unfortunately, it is not possible to simply look at a new text-classification task and immediately have a good idea of whether it has a large margin. The margin property is observable only after training data becomes available and requires training the SVM. To overcome this problem, this second step connects the large-margin property with more intuitive and more observable properties of text-classification tasks. It will show how the properties identified in Section 4.1 necessarily lead to a large margin. This explains why and when SVMs give good performance on text-classification task despite the high-dimensional feature space.

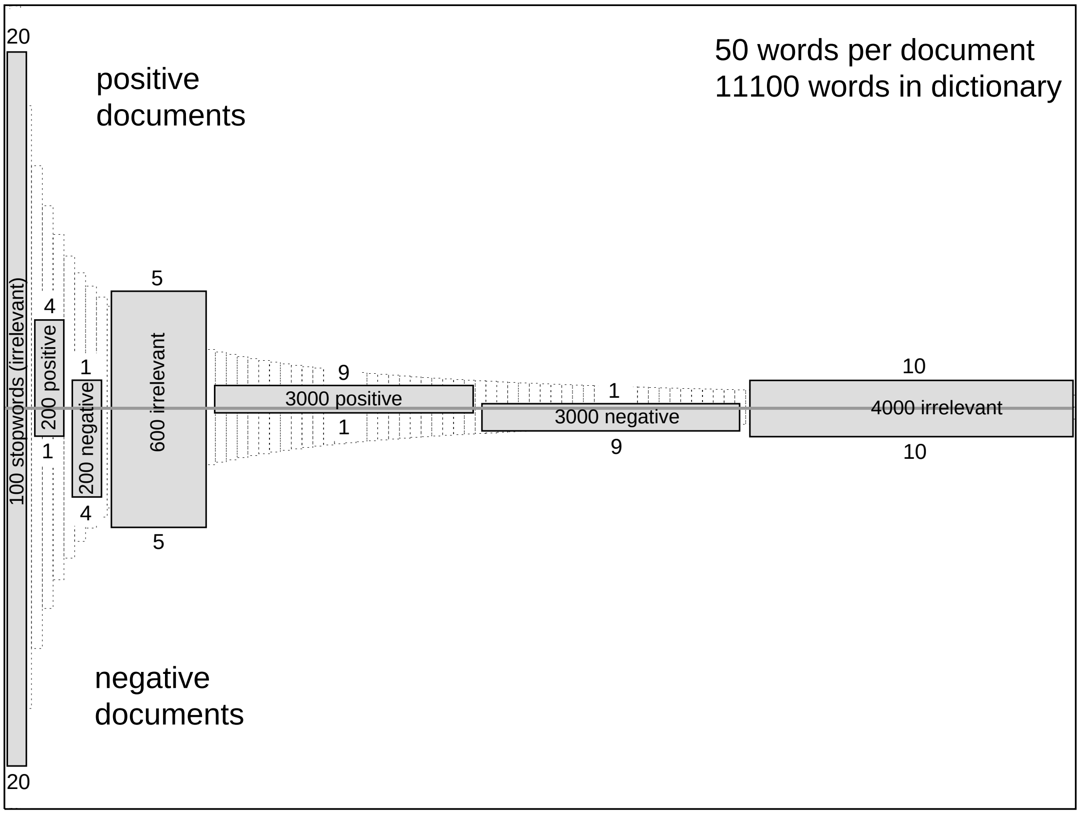


Figure 4.5: A simple example of a TCat-concept.

4.2.2.1 Homogeneous TCat-Concepts

Figure 4.5 depicts a stereotypical text classification task that I will use to motivate the model developed in the following. More sophisticated versions of this kind of concept will later be shown to model real text-classification tasks. Figure 4.5 illustrates the "word-frequency histogram" of a binary classification problem with the two classes POS and NEG. Documents from the two classes differ by the frequency with which certain words occur in them.

- Independently of whether a document is from the positive or the negative class, each document contains 20 stopwords from a set of 100 such words.

- There are 1,000 medium-frequency words. From a subset of 600 such words, again both positive and negative documents contain 5 occurrences. But there are also two groups of 200 words each that occur primarily in positive or negative documents, respectively. In particular, from one group there are 4 in each positive document and only 1 in each negative document. Respectively, from the other group there are 4 occurrences in each negative document while there is only one in each positive document.

- Similarly, for the set of 10,000 low-frequency words, there is a subset of 4,000 words of which there are 10 occurences in both positive and negative documents. But there are two sets of 3,000 words each that occur primarily in positive or negative documents with a frequency of 9 versus 1.

In how far does this example resemble the properties of text-classification tasks identified in Section 4.1?

High-Dimensional Input Space: There are 11,100 features, which is on the same order of magnitude as real text-classification tasks.

Sparse Document Vectors: Each document is only 50 words long, which means there are at least 11,050 zero entries in each document vector.

High Level of Redundancy: In each document there are 4 medium-frequency words and 9 low-frequency words that indicate the class of the document. Considering the document length of 50 words, this is a fairly high level of redundancy.

Heterogenous Use of Terms: For both positive and negative documents there are a group of 200 medium-frequency words and a group of 3,000 low-frequency words. From each group there can be an arbitrary subset of 4 for the medium-frequency words and 9 for the low-frequency words in each document. Considering only the medium-frequency words, this implies that there can be 50 documents in the same class that do not share a single medium-frequency term from this group. This mimics the property of text classification tasks identified in Section 4.1.2.

Zipf's Law: There is a small number of words (100 stopwords) that occur very frequently, a set of 1,000 words of medium frequency, and a large set of 10,000 low-frequency words. This does resemble Zipf's law.

Justifying the validity of this type of target concept as a model of text-classification tasks in more detail is subject of the next section.

The key observation for the example in Figure 4.5 is that it is linearly separable with a lower-bounded margin. Let documents be represented by term-frequency vectors and define a hyperplane classifier

$$h(\vec{x}) = \vec{w} \cdot \vec{x} + b = \sum_{i=1}^{11100} w_i x_i + b \tag{4.14}$$

with $b = 0$ and

$$w_i = \begin{cases} +0.23 & \text{for the 200 medium-frequency words indicating POS} \\ -0.23 & \text{for the 200 medium-frequency words indicating NEG} \\ +0.04 & \text{for the 3000 low-frequency words indicating POS} \\ -0.04 & \text{for the 3000 low-frequency words indicating NEG} \\ 0 & \text{for all other words} \end{cases} \tag{4.15}$$

It is easy to verify that the hyperplane defined this way has a margin δ of at least $\sqrt{1/30.15}$ for the example in Figure 4.5. This example gives a first hint towards how the bound from Theorem 1 leads to learnability results for text-classification with SVMs. Before we can analyze this in detail, we need to abstract from this particular example to a parameterized model that can describe text-classification tasks more generally.

Definition 8 (Homogenous TCat-Concepts) *The TCat-concept*

$$TCat([p_1 : n_1 : f_1], ..., [p_s : n_s : f_s]) \tag{4.16}$$

describes a binary classification task with s disjoint sets of features. The i-th set includes f_i features. Each positive example contains p_i occurrences of features from the respective set, and each negative example contains n_i occurrences. The same feature can occur multiple times in one document.

	high frequency	medium frequency	low frequency
pos	**98 words** all any assignment assignments available be book c chapter class code course cse description discussion document due each eecs exam exams fall final ... section set should solution solutions spring structures students syllabus ta text textbook there thursday topics tuesday unix use wednesday week will you your	**431 words** account acrobat adapted addison adt ahead aho allowed alternate announced announcement announcements answers appointment approximately ... tuesdays turing turn turned tuth txt uidaho uiowa ullman understand ungraded units unless upenn usr vectors vi walter weaver wed wednesdays weekly weeks weights wesley yurttas	**5045 words** 002cc 009a 00a 00om 01oct 01pm 02pm 03oct 03pm 03sep 04dec ... gradable gradebook gradebooks gradefreq1 gradefreq2 gradefreq3 graders gradesheet gradients grafica grafik ... zimmermann zinc zipi zipser zj zlocate znol zoran zp zwatch zwhere zwiener zyda
neg	acm address am austin ca california center college computational conference contact current currently d department dr faculty fax graduate group he ... me member my our parallel performance ph pp proceedings professor publications recent research sciences support technical technology university vision was working **52 words**	aaai academy accesses accurate adaptation advisor advisory affiliated affiliations agent agents alberta album alumni amanda america amherst annual ... victoria virginia visiting visitors visualization vita vitae voice wa watson weather webster went west wi wife wireless wisconsin worked workshop workshops wrote yale york **341 words**	0a 0b 0b1 0e 0f 0r 0software 0x82d4ff 100k 100mhz 100th 1020x620 102k 103k ... lunar lunches lunchtime lund lundberg lunedi lung luniewski luo luong lupin lupton lure lurker lus ... zuo zuowei zurich zvi zw zwaenepoel zwarico zwickau zwilling zygmunt zzhen00 **24276 words**
	high frequency	**medium frequency**	**low frequency**

Figure 4.6: Indicative words for the WebKB category "course" partitioned by occurrence frequency.

While this definition does not include noise (e.g. on the number of features from each set), Section 4.5 shows how this assumption can be relaxed in a straightforward way. Applying this definition to the example in Figure 4.5, it is easy to verify that the example can be described as a

$$\begin{array}{lll} TCat(& [20:20:100], & \text{\# high frequency} \\ & [4:1:200],[1:4:200],[5:5:600], & \text{\# medium frequency} \\ & [9:1:3000],[1:9:3000],[10:10:4000] & \text{\# low frequency} \\ ) & & \end{array} \tag{4.17}$$

concept. While this is an artificial example, is it possible to model real text-classification tasks as TCat-concepts?

4.2.2.2 Experimental Validation

This section analyzes text-classification tasks from the Reuters, the WebKB, and the Ohsumed collection (see Section 2.7.1). It shows that they can be modeled as TCat-concepts.

Let us start with the category "course" from the WebKB collection. First, we need to partition the feature space into disjoint sets of positive indicators, negative indicators, and irrelevant features. Using the simple strategy of selecting features by their odds ratio, there are 98 high-frequency words that indicate positive documents (odds ratio greater

	high frequency		medium frequency		low frequency		
	98 pos.	52 neg.	431 pos.	341 neg.	5,045 pos.	24,276 neg.	8116 rest
pos. doc.	27.7%	1.5%	5.9%	0.3%	3.2%	0.2%	61.2%
neg. doc.	10.4%	7.7%	0.7%	4.4%	0.2%	7.7%	68.9%

Table 4.2: Composition of an "average" positive and an "average" negative document with respect to the WebKB category "course". The numbers give the percentage of occurrences that are due to words from the groups identified in Figure 4.6.

than 2) and 52 high-frequency words indicating negative documents (odds ratio less than 0.5). An excerpt of these words is given in Figure 4.6. Similarly, there are 431 (341) medium-frequency words that indicate positive (negative) documents with an odds ratio greater than 5 (less than 0.2). In the low-frequency spectrum there are 5,045 positive indicators (odds ratio greater than 10) and 24,276 negative indicators (odds ratio less than 0.1). All other words in the vocabulary are assumed to carry no information.

To abstract from the details of particular documents, it is useful to define what a typical document for this task looks like. In some sense, an "average" document captures what's typical. An average WebKB document is 277 words long. For positive examples of the category "course", on average 27.7% of the 277 occurrences come from the set of 98 high-frequency positive indicators while these words account for only 10.4% of the occurrences in an average negative document. The relative occurrence frequencies for the other word groups are given in Table 4.2. Applying these percentages to the average document length, this table can be directly translated into the following TCat-concept.

$$
\begin{array}{lll}
TCat_{course}(& [77:29:98],[4:21:52], & \text{\# high frequency} \\
 & [16:2:431],[1:12:341], & \text{\# medium frequency} \\
 & [9:1:5045],[1:21:24276], & \text{\# low frequency} \\
 & [169:191:8116] & \text{\# rest} \\
) & &
\end{array}
\tag{4.18}
$$

This shows that the text-classification task connected with the WebKB category "course" can be modeled as a TCat-concept, if one assumes that documents are of homogeneous length and composition. Section 4.5 will show how this assumption of homogeneity can be relaxed.

Similar TCat-concepts can also be found for other tasks. For the Reuters category "earn" the same procedure leads to the TCat-concept

$$
\begin{array}{lll}
TCat_{earn}(& [33:2:65],[32:65:152], & \text{\# high frequency} \\
 & [2:1:171],[3:21:974], & \text{\# medium frequency} \\
 & [3:1:3455],[1:10:17020], & \text{\# low frequency} \\
 & [78:52:5821] & \text{\# rest} \\
) & &
\end{array}
\tag{4.19}
$$

as an average case model. The model for the Ohsumed category "pathology" is

$$
\begin{array}{lll}
TCat_{pathology}(& [2:1:10],[1:4:22], & \text{\# high frequency} \\
 & [2:1:92],[1:2:94], & \text{\# medium frequency} \\
 & [5:1:4080],[1:10:20922], & \text{\# low frequency} \\
 & [197:190:13459] & \text{\# rest} \\
). & &
\end{array}
\tag{4.20}
$$

For these formal models it is now possible to prove learnability properties.

4.2.2.3 Learnability

This section provides the connection between the properties of TCat-concepts and the bound for the generalization performance of a support vector machine. The first lemma shows that homogenous TCat-concepts are generally separable with a certain margin. Using the fact that term frequencies obey Zipf's law, a second lemma shows that the Euclidian length of document vectors is small for text-classification tasks. These two results lead to the main learnability result for TCat-concepts.

Lemma 2 (Lower Bound on the Margin of Noise-Free TCat-Concepts) *For $TCat([p_1 : n_1 : f_1], ..., [p_s : n_s : f_s])$-concepts, there is always a hyperplane passing through the origin that has a margin δ bounded by*

$$\delta^2 \geq \frac{ac - b^2}{a + 2b + c} \qquad \text{with} \qquad \begin{aligned} a &= \sum_{i=1}^{s} \frac{p_i^2}{f_i} \\ b &= \sum_{i=1}^{s} \frac{p_i n_i}{f_i} \\ c &= \sum_{i=1}^{s} \frac{n_i^2}{f_i} \end{aligned} \tag{4.21}$$

Proof *Define $\vec{p}^T = (p_1, \ldots, p_s)^T$ and $\vec{n}^T = (n_1, \ldots, n_s)^T$, as well as the diagonal matrix F with $f_1, \ldots, f_s$ on the diagonal.*

The margin of the maximum-margin hyperplane that separates a given training sample $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$ and passes through the origin can be derived from the solution of following optimization problem.

$$W(\vec{w}) = min \frac{1}{2} \vec{w}^T \vec{w} \tag{4.22}$$

$$\begin{aligned} s.t. \quad & y_1[\vec{x}_1^T \vec{w}] \geq 1 \\ & \vdots \\ & y_n[\vec{x}_n^T \vec{w}] \geq 1 \end{aligned} \tag{4.23}$$

The hyperplane corresponding to the solution vector $\vec{w}^*$ has a margin $\delta = \frac{1}{\sqrt{2W(\vec{w}^*)}}$. By adding constraints to this optimization problem, it is possible to simplify its solution and get a lower bound on the margin. Let us add the additional constraint that within each group of f_i features the weights are required to be identical. Then $\vec{w}^T \vec{w} = \vec{v}^T F \vec{v}$ for a vector $\vec{v}$ of dimensionality s. The constraints (4.23) can also be simplified. By definition, each example contains a certain number of features from each group. This means that all constraints for positive examples are equivalent to $\vec{p}^T \vec{v} \geq 1$ and, respectively, $\vec{n}^T \vec{v} \leq -1$ for the negative examples. This leads to the following simplified optimization problem.

$$W'(\vec{v}) = min \frac{1}{2} \vec{v}^T F \vec{v} \tag{4.24}$$

$$s.t. \quad \vec{p}^T \vec{v} \geq 1 \tag{4.25}$$

$$\vec{n}^T \vec{v} \leq -1 \tag{4.26}$$

Let $\vec{v}^*$ be the solution. Since $W'(\vec{v}^*) \geq W(\vec{w}^*)$, $\delta \geq \frac{1}{\sqrt{2W'(\vec{v}^*)}}$ is a lower bound for the margin. It remains to find an upper bound for $W'(\vec{v}^*)$ that can be computed in closed form. Introducing Lagrange multipliers, the solution $W'(\vec{v}^*)$ equals the value $L(\vec{v}, \alpha_+, \alpha_-)^*$ of

$$L(\vec{v}, \alpha_+, \alpha_-) = \frac{1}{2} \vec{v}^T F \vec{v} - \alpha_+ (\vec{p}^T \vec{v} - 1) + \alpha_- (\vec{n}^T \vec{v} + 1) \tag{4.27}$$

at its saddle-point. $\alpha_+ \geq 0$ *and* $\alpha_- \geq 0$ *are the Lagrange multipliers for the two constraints (4.25) and (4.26). Using the fact that*

$$\frac{dL(\vec{v}, \alpha_+, \alpha_-)}{d\vec{v}} = 0 \tag{4.28}$$

at the saddle point one gets a closed form solution for $\vec{v}$.

$$\vec{v} = F^{-1}\left[\alpha_+\vec{p} - \alpha_-\vec{n}\right] \tag{4.29}$$

For ease of notation one can equivalently write

$$\vec{v} = F^{-1}XY\vec{\alpha} \tag{4.30}$$

with $X = (\vec{p}, \vec{n})$, $Y = diag(1, -1)$, *and* $\vec{\alpha}^T = (\alpha_+, \alpha_-)$ *appropriately defined. Substituting into the Lagrangian results in*

$$L(\vec{\alpha}) = 1^T\vec{\alpha} - \frac{1}{2}\vec{\alpha}^T YX^T F^{-1}XY\vec{\alpha} \tag{4.31}$$

To find the saddle points one has to maximize this function over $\vec{\alpha}^T = (\alpha_+, \alpha_-)^T$ *subject to* $\alpha_+ \geq 0$ *and* $\alpha_- \geq 0$. *Since only a lower bound on the margin is needed, it is possible to drop the constraints* $\alpha_+ \geq 0$ *and* $\alpha_- \geq 0$. *Removing the constraints can only increase the objective function at the solution. So the unconstrained maximum* $L'(\vec{\alpha})^*$ *is greater or equal to* $L(\vec{\alpha})^*$. *Setting the derivative of (4.31) to 0*

$$\frac{dL'(\vec{\alpha})}{d\vec{\alpha}} = 0 \quad\Leftrightarrow\quad \vec{\alpha} = (YX^TF^{-1}XY)^{-1}1 \tag{4.32}$$

and substituting into (4.31) yields the unconstrained maximum:

$$L'(\vec{v}, \vec{\alpha})^* = \frac{1}{2}1^T(YX^TF^{-1}XY)^{-1}1 \tag{4.33}$$

The special form of $(YX^TF^{-1}XY)$ *makes it possible to compute its inverse in closed form.*

$$(YX^TF^{-1}XY)^{-1} = \begin{pmatrix} \vec{p}^TF^{-1}\vec{p} & -\vec{p}^TF^{-1}\vec{n} \\ -\vec{n}^TF^{-1}\vec{p} & \vec{n}^TF^{-1}\vec{n} \end{pmatrix}^{-1} \tag{4.34}$$

$$= \begin{pmatrix} a & -b \\ -b & c \end{pmatrix}^{-1} \tag{4.35}$$

$$= \frac{1}{ac - b^2}\begin{pmatrix} a & b \\ b & c \end{pmatrix} \tag{4.36}$$

Substituting into (4.33) completes the proof. ▮

Separability implies that the training loss is zero. So we only still need to bound the maximum Euclidian length R of document vectors before it is possible to apply Theorem 1. Clearly, the document vector of a document with l words cannot have a Euclidian length greater than l. Nevertheless, this bound is very loose for real document vectors. To bound the quantity R more tighly it is possible to make use of Zipf's law.

Assume that the term frequencies in every document follow the generalized Zipf's law

$$TF_r = \frac{c}{(r + k)^\phi} \tag{4.37}$$

This assumption about Zipf's law does not imply that a particular word occurs with a certain frequency in every document. It is much weaker; it merely implies that the r-th most frequent word occurs with a particular frequency. This r-th most frequent word can be different in different documents, in particular depending on the class the document comes from. The following lemma connects the length of the document vectors to Zipf's law.

Lemma 3 (Euclidian Length of Document Vectors) *If the ranked term frequencies TF_r in a document with l terms have the form of the generalized Zipf's law*

$$TF_r = \frac{c}{(r+k)^\phi} \tag{4.38}$$

based on their frequency rank r, then the squared Euclidian length of the document vector $\vec{x}$ of term frequencies is bounded by

$$||\vec{x}|| \leq \sqrt{\sum_{r=1}^{d} \left(\frac{c}{(r+k)^\phi}\right)^2} \quad \textit{with } d \textit{ such that } \sum_{r=1}^{d} \frac{c}{(r+k)^\phi} = l \tag{4.39}$$

Proof *From the connection between the frequency rank of a term and its absolute frequency it follows that the r-th most frequent term occurs*

$$TF_r = \frac{c}{(r+k)^\phi} \tag{4.40}$$

times. The document vector $\vec{x}$ has d non-zero entries which are the values $TF_1, ..., TF_d$. Therefore, the Euclidian length of the document vector $\vec{x}$ is

$$\vec{x}^T\vec{x} = \sum_{r=1}^{d} \left(\frac{c}{(r+k)^\phi}\right)^2 \tag{4.41}$$

▮

The fact that the term frequencies follow some form of Zipf's law has a strong impact on the learnability of text-classification tasks. Zipf's law implies that most terms do not repeat often and that the number of distinct terms d is high. If Zipf's law did not hold, a single word could repeat l times, leading to a document vector with Euclidian length l. Instead, Zipf's law leads to comparably short document vectors and implies a small value of R^2 in the bound on the expected generalization performance.

Combining Lemma 2 and Lemma 3 with Theorem 1 leads to the following main result.

Theorem 2 (Learnability of TCat-Concepts) *For $TCat([p_1 : n_1 : f_1], ..., [p_s : n_s : f_s])$-concepts and documents with l terms distributed according to the generalized Zipf's law $TF_r = \frac{c}{(r+k)^\phi}$, the expected generalization error of an (unbiased) SVM after training on n examples is bounded by*

$$\mathcal{E}(Err^n(h_{SVM})) \leq \rho \frac{R^2}{n+1} \frac{a+2b+c}{ac-b^2} \qquad \textit{with} \qquad \begin{array}{l} a = \sum_{i=1}^{s} \frac{p_i^2}{f_i} \\ b = \sum_{i=1}^{s} \frac{p_i n_i}{f_i} \\ c = \sum_{i=1}^{s} \frac{n_i^2}{f_i} \\ R^2 = \sum_{r=1}^{d} \left(\frac{c}{(r+k)^\phi}\right)^2 \end{array} \tag{4.42}$$

unless $\forall_{i=1}^{s} : p_i = n_i$. d is chosen so that $\sum_{r=1}^{d} \frac{c}{(r+k)^\phi} = l$. For unbiased SVMs ρ equals 1, and for biased SVMs ρ equals 2.

Proof *Using the fact that TCat-concepts are separable (and therefore stable), if at least for one i the value of p_i is different from n_i, the result from Theorem 1 reduces to*

$$\mathcal{E}(Err^n(h_{SVM})) \leq \frac{1}{n+1}\,\rho\, E\!\left(\frac{R^2}{\delta^2}\right) \tag{4.43}$$

since all ξ_i are zero for a sufficiently large value of C. Lemma 2 gives a lower bound for δ^2 which can be used to bound the expectation

$$E\!\left(\frac{R^2}{\delta^2}\right) \leq \rho\frac{a+2b+c}{ac-b^2}\, E\!\left(R^2\right) \tag{4.44}$$

It remains for us to give an upper bound for $E(R^2)$. R^2 is the maximum Euclidian length of any feature vector in the training data. Since the term frequencies in each example follow the generalized Zipf's law $TF_r = \frac{c}{(r+k)^{\phi}}$, it is possible to use Lemma 3 to bound R^2 and therefore $E(R^2)$. ▮

4.3 Comparing the Theoretical Model with Experimental Results

The previous sections formally prove that a large expected margin with low training error leads to a low expected prediction error. Furthermore, they prove how margin is related to the properties of TCat-concepts, and experimentally verify that real text-classification tasks can be modeled with TCat-concepts. This section verifies not only that the individual steps are well justified, but also that their conjunction produces meaningful results. To show this, this section compares the generalization performance as predicted by the model with the generalization performance found in experiments.

In Section 4.2.2.2 a TCat-model for the WebKB category "course" was estimated to be

$$\begin{array}{lll} TCat_{course}(& [77:29:98],[4:21:52], & \#\text{ high frequency}\\ & [16:2:431],[1:12:341], & \#\text{ medium frequency}\\ & [9:1:5045],[1:21:24276], & \#\text{ low frequency}\\ & [169:191:8116] & \#\text{ rest}\\ ). & & \end{array} \tag{4.45}$$

An average document in the WebKB collection is 277 words long and contains 130 distinct terms. Making the assumption that documents are sufficiently homogeneous, the generalized Zipf's law estimated for the full WebKB collection in Section 4.1.5 also applies to individual documents after rescaling the multiplicative constant appropriately. The resulting bound for R^2 is

$$\sum_{i=1}^{130} TF_i^2 = 1899.7 \geq R^2 \tag{4.46}$$

Substituting the values into the bound from Theorem 2 leads to the following bound on the expected error.

$$\mathcal{E}(Err^n(h_{SVM})) \leq \frac{0.2331 \cdot 1899.7}{n+1} \leq \frac{443}{n+1} \tag{4.47}$$

	model	experiment	
	$\mathcal{E}(Err^n(h_{SVM}))$	$Err^n_{test}(h_{SVM})$	PRBEP
WebKB "course"	11.2%	4.4%	92.4
Reuters "earn"	1.5%	1.3%	98.1
Ohsumed "pathology"	94.5%	23.1%	49.0

Table 4.3: Comparing the expected error predicted by the model with the error rate and the precision/recall breakeven point on the test set for the WebKB category "course", the Reuters category "earn", and the Ohsumed category "pathology" with TF weighting and $C = 1000$. No stopword removal and no stemming are used.

n denotes the number of training examples. Consequently, after training on 3957 examples the model predicts an expected generalization error of less than 11.2%.

An analog procedure for the Reuters category "earn" leads to the bound

$$\mathcal{E}(Err^n(h_{SVM})) \leq \frac{0.1802 \cdot 762.9}{n+1} \leq \frac{138}{n+1} \tag{4.48}$$

so that the expected generalization error after 9603 training examples is less than 1.5%. Similarly, the bound for the Ohsumed category "pathology" is

$$\mathcal{E}(Err^n(h_{SVM})) \leq \frac{7.4123 \cdot 1275.8}{n+1} \leq \frac{9457}{n+1}, \tag{4.49}$$

leading to an expected generalization error of less than 94.5% after 10,000 training examples.

Table 4.3 compares the expected generalization error predicted by the model with the generalization performance observed in experiments. While it is unreasonable to expect that the model precisely predicts the exact performance observed on the test set, Table 4.3 shows that the model captures which classification tasks are more difficult than others. In particular, it does correctly predict that "earn" is the easiest task, "course" is the second easiest task, and that "pathology" is the most difficult one. This shows that TCat-concepts can sufficiently formalize the key properties of text-classification tasks that are responsible for learnability with support vector machines.

4.4 Sensitivity Analysis: Difficult and Easy Learning Tasks

The previous section revealed that the bound on the expected generalization error can be large for some TCat-concepts while it is small for others. The following shows that the occurrence frequency, the level of redundancy, and the discriminatory power of term sets are primary influence factors.

4.4.1 Influence of Occurrence Frequency

The following TCat-concept has all discriminative features in the high-frequency range, while the medium and low-frequency words do not help distinguish between classes.

$$\begin{array}{lll} TCat_{hf}(& [16:4:10],[4:16:10],[20:20:30], & \text{\# high frequency} \\ & [30:30:2000], & \text{\# medium frequency} \\ & [30:30:30000] & \text{\# low frequency} \\ \quad) & & \end{array} \tag{4.50}$$

	$Err^n_{test}(h_{SVM})$		PRBEP	
	tf	tfidf	tf	tfidf
WebKB “course”	4.4%	7.5%	92.4	84.1
Reuters “earn”	1.3%	1.3%	98.1	98.1
Ohsumed “pathology”	23.1%	21.1%	49.0	52.0

Table 4.4: Precision/recall breakeven point and error rate on the test set for the WebKB category “course”, the Reuters category “earn”, and the Ohsumed category “pathology” with and without TFIDF weighting. No stopword removal and no stemming are used.

Analogously, the TCat-concept

$$\begin{array}{lll} TCat_{mf}(& [40:40:50], & \#\text{ high frequency} \\ & [12:3:500],[3:12:500],[15:15:1000], & \#\text{ medium frequency} \\ & [30:30:30000] & \#\text{ low frequency} \\ ) & & \end{array} \tag{4.51}$$

has discriminative features only in the medium-frequency range, and the concept

$$\begin{array}{lll} TCat_{lf}(& [40:40:50], & \#\text{ high frequency} \\ & [30:30:2000], & \#\text{ medium frequency} \\ & [12:3:7500],[3:12:7500],[15:15:15000] & \#\text{ low frequency} \\ ) & & \end{array} \tag{4.52}$$

has discriminative features only in the low-frequency range. Assuming that the frequency distribution of words for all three tasks follows the same generalized Zipf's law, then Theorem 2 ensures the lowest generalization error for the task $TCat_{hf}$ with discriminative features in the high-frequency range. The margin δ for $TCat_{hf}$ is always greater than 2.68, while the margin of $TCat_{mf}$ and $TCat_{lf}$ are only greater than 0.28 and 0.07 respectively. This suggests that discriminative features particularly in the high-frequency range can lead to a low generalization error.

For tasks with discriminative features mostly in the low-frequency range, TFIDF weighting can improve performance compared to a plain TF representation as above. The effect of TFIDF weighting is an increase of the relative weight of low-frequency terms. The term frequencies are multiplied by a factor that is largest for low-frequency terms and smallest for high-frequency terms. A simplified approximation to TFIDF weighting is the following. The term frequencies of low-frequency words are multiplied by a factor of 2 while all other term frequencies remain unscaled. Applying this weighting to $TCat_{lf}$ leads to

$$\begin{array}{lll} TCat_{lf-tfidf}(& [40:40:50], & \#\text{ high frequency} \\ & [30:30:2000], & \#\text{ medium frequency} \\ & [24:6:7500],[6:24:7500],[30:30:15000] & \#\text{ low frequency} \\ ). & & \end{array} \tag{4.53}$$

After correcting for an increased length of the document vector R^2 by a factor of 1.9, the margin of $TCat_{lf-tfidf}$ with TFIDF weighting is larger than 0.09 compared to only 0.07 for $TCat_{lf}$ without TFIDF weighting. On the other hand, applying the same weighting scheme to $TCat_{hf}$ leads to a decreased bound on the margin. Here, the margin of 2.67 for $TCat_{hf}$ compares to only 1.94 for $TCat_{hf-tfidf}$. This suggests that TFIDF weighting is beneficial for those learning tasks with discriminative features primarily in the low-frequency region. Such a task is the Ohsumed category "pathology", while "earn" and "course" have many discriminative features also in the high and medium-frequency range. Table 4.4 shows that the predicted effect of TFIDF weighting can be observed in experiments. The table compares error rate and precision/recall breakeven points for representations with and without TFIDF weighting. While TFIDF weighting does improve performance for the Ohsumed category, it does not help or even hurts performance for the other two tasks. This again provides evidence for the validity of the model and gives insight into when the use of TFIDF weighting is appropriate.

4.4.2 Discriminative Power of Term Sets

Intuitively, the extent to which vocabulary differs between classes should make a difference for learnability. Is there a set of words that occurs only in positive documents or only in negative documents? Or, more weakly, is it that certain groups of words merely occur more or less frequently with respect to the class? The following three tasks differ in this respect, which can be interpreted as the discriminative power of term sets. The TCat-concept

$$\begin{array}{lll} TCat_{ld}(& [40:40:50], & \#\text{ high frequency} \\ & [9:6:500],[6:9:500],[15:15:1000], & \#\text{ medium frequency} \\ & [9:6:7500],[6:9:7500],[15:15:15000] & \#\text{ low frequency} \\ ) & & \end{array} \tag{4.54}$$

has term sets with low discriminative power, since the occurrence ratios are only 9 vs. 6. The discriminative power is higher for the concept

$$\begin{array}{lll} TCat_{md}(& [40:40:50], & \#\text{ high frequency} \\ & [12:3:500],[3:12:500],[15:15:1000], & \#\text{ medium frequency} \\ & [12:3:7500],[3:12:7500],[15:15:15000] & \#\text{ low frequency} \\ ) & & \end{array} \tag{4.55}$$

and it is highest for

$$\begin{array}{lll} TCat_{hd}(& [40:40:50], & \#\text{ high frequency} \\ & [15:0:500],[0:15:500],[15:15:1000], & \#\text{ medium frequency} \\ & [15:0:7500],[0:15:7500],[15:15:15000] & \#\text{ low frequency} \\ ). & & \end{array} \tag{4.56}$$

The discriminative power is reflected in the bound for the margin from Lemma 2. The bound is 0.48 for the margin of $TCat_{hd}$, while it is only 0.29 for $TCat_{md}$, and 0.09 for $TCat_{ld}$. With an R^2 constant over all tasks, Theorem 2 ensures the best generalization performance for the task with highly discriminative term sets.

4.4.3 Level of Redundancy

The final important factor for ensuring a low expected generalization error is the level of redundancy in the TCat-concept. The following three TCat-concepts have discriminative features only in the medium-frequency range, but differ in how many cues there are in each document. For

$$\begin{array}{lll} TCat_{lr}(& [40:40:50], & \#\text{ high frequency} \\ & [3:0:100],[0:3:100],[27:27:1800], & \#\text{ medium frequency} \\ & [30:30:30000] & \#\text{ low frequency} \\ ) & & \end{array} \tag{4.57}$$

most medium-frequency terms do not help differentiate between the classes. Each document contains only three words that indicate the positive or the negative class respectively. This can be interpreted as a low level of redundancy. Similarly, the task

$$\begin{array}{lll} TCat_{mr}(& [40:40:50], & \#\text{ high frequency} \\ & [15:0:500],[0:15:500],[15:15:1000], & \#\text{ medium frequency} \\ & [30:30:30000] & \#\text{ low frequency} \\ \quad) & & \end{array} \tag{4.58}$$

provides a medium level of redundancy, and the concept

$$\begin{array}{lll} TCat_{hr}(& [40:40:50], & \#\text{ high frequency} \\ & [30:0:1000],[0:30:1000], & \#\text{ medium frequency} \\ & [30:30:30000] & \#\text{ low frequency} \\ \quad) & & \end{array} \tag{4.59}$$

provides a high level of redundancy. Using the result from Lemma 2 the largest margin is assured for the highly redundant TCat-concept $TCat_{hr}$ with δ greater than 0.67, while for $TCat_{mr}$ the margin is only greater than 0.47, and for $TCat_{lr}$ it is greater than 0.21. Assuming again that R^2 is the same for all three tasks, Theorem 2 ensures the best generalization performance for the high redundancy task. This suggests that a high level of redundancy is a desirable property of text-classification tasks.

4.5 Noisy TCat-Concepts

While the homogenous TCat-concepts considered so far allow for large variability within feature sets, they strictly prescribe the relative word frequencies between feature sets. Clearly, the constraint that each document contains an exact number of occurrences from each word set of the TCat-concept is very restrictive. In practice, the occurrence frequencies will vary among documents to some extent. In the example task from Figure 4.5, for some positive example there may occur only three medium-frequency positive indicators instead of the four indicators described in the model. This type of noise will be called *attribute noise*. A second form of noise present in almost every text-classification task is *classification noise*. A positive document could be mislabeled as a negative example or vice versa.

Both types of noise can be incorporated into the model. The following argument shows that the learning bounds scale smoothly with the amount of noise. For both attribute noise and classification noise, the increase of the error bound from Theorem 2 is limited.

In the case of attribute noise, two kinds of deviations move positive and negative examples closer together:

- a positive (negative) document d_i can contain $\Delta_{j,i}$ fewer positive (negative) indicators from a word set s_j than prescribed by the model
- a positive (negative) document d_i can contain $\Delta_{j,i}$ more negative (positive) indicators from a word set s_j than prescribed by the model

In the first case the distance to the hyperplane will decrease proportional to $\Delta_j \cdot w_j$, if w_j is the value in the weight vector for the words in the set s_j. Intuitively, the document lacks Δ_j words to push the document vector further to the "correct" side of the margin. Similarly in the second case, the distance will decrease proportional to $\Delta_j \cdot w_j$, if again w_j is the value in the weight vector for the set s_j. Here, the additional counter-indicative words pull the document vector to the "wrong" side of the margin. Knowing the expectation of $\vec{\Delta}^T = (\Delta_1, \ldots, \Delta_s)^T$ with $\Delta_j = \sum_{i=1}^{n+1} \Delta_{j,i}$ for training sets of size $n+1$ is sufficient to limit the effect of noise on the expected error bound compared to the noise-free case.

Theorem 3 (Learnability of TCat-Concepts with Attribute Noise) *Denote with $< \vec{w}_{sep} >$ the unbiased maximum margin hyperplane that separates the noise-free TCat-concept according to Lemma 2. If this TCat-concept is distorted by attribute noise according to $\mathcal{E}(\vec{\Delta})$, then the expected error of this noisy TCat-concept does not exceed the expected error bound of the noise-free TCat-concept by more than*

$$2\, \vec{w}_{sep}^T\, \mathcal{E}(\vec{\Delta}) \tag{4.60}$$

for $C = \frac{1}{\rho R^2}$ and R^2 fixed.

Proof *Denote with $< \vec{w}_{noise}, \vec{\xi}_{noise} >$ the solution of the primal optimization problem for some training set S of size $n+1$ drawn according to the noisy TCat-concept. Let $\vec{\Delta}_i$ be the noise vectors for the examples in the current training set. Starting from the solution of the noise-free separator $< \vec{w}_{sep} >$ it is possible to construct a feasible point of the primal optimization problem for the noisy training set.*

$$\frac{1}{2}\vec{w}_{sep}^T\vec{w}_{sep} + C\sum_{i=1}^{n+1} \vec{\Delta}_i^T \vec{w}_{sep} \geq \frac{1}{2}\vec{w}_{noise}^T\vec{w}_{noise} + C\sum_{i=1}^{n+1} \xi_{i,noise} \tag{4.61}$$

It follows from the proof of Lemma 1 that

$$Err_{loo}^{n+1}(S) \leq \frac{\rho R^2}{n+1}\left[\vec{w}_{noise}^T\vec{w}_{noise} + C\sum_{i=1}^{n} \xi_{i,noise}\right] \tag{4.62}$$

This quantity can be upper bounded as follows.

$$\begin{aligned} Err_{loo}^{n+1}(S) &\leq \frac{\rho R^2}{n+1}\left[\vec{w}_{noise}^T\vec{w}_{noise} + C\sum_{i=1}^{n} \xi_{i,noise}\right] &(4.63)\\ &\leq \frac{\rho R^2}{n+1}\left[\vec{w}_{noise}^T\vec{w}_{noise} + 2C\sum_{i=1}^{n} \xi_{i,noise}\right] &(4.64)\\ &\leq \frac{\rho R^2}{n+1}\left[\vec{w}_{sep}^T\vec{w}_{sep} + 2C\sum_{i=1}^{n} \vec{\Delta}_i^T\vec{w}_{sep}\right] &(4.65)\\ &\leq \frac{\rho R^2}{n+1}\left[\vec{w}_{sep}^T\vec{w}_{sep} + 2C\vec{w}_{sep}^T\sum_{i=1}^{n} \vec{\Delta}_i\right] &(4.66) \end{aligned}$$

Note that $\vec{w}_{sep}$ has zero variance according to the construction in Lemma 2. Taking the expectation gives an upper bound on the expected error

$$\mathcal{E}(Err^n(h_{SVM})) \leq \frac{\rho R^2}{n+1}\left[\vec{w}_{sep}^T\vec{w}_{sep} + 2C(n+1)\vec{w}_{sep}^T\mathcal{E}(\Delta)\right] \tag{4.67}$$

which differs from the bound for the noise-free TCat-concept

$$\mathcal{E}(Err^n(h_{SVM})) \leq \frac{\rho R^2}{n+1}\vec{w}_{sep}^T\vec{w}_{sep} \tag{4.68}$$

by

$$2\rho R^2 C\vec{w}_{sep}^T\mathcal{E}(\Delta) \tag{4.69}$$

∎

Similarly, classification noise also leads to a smooth increase of the bound on the expected error rate. It can be integrated in the following way.

Theorem 4 (Learnability of TCat-Concepts with Classification Noise) *Denote with* $< \vec{w}_{sep} >$ *the unbiased maximum margin hyperplane that separates the noise-free TCat-concept according to Lemma 2. If this TCat-concept is distorted by classification noise with a rate of* Δ*, then the expected error of this noisy TCat-concept does not exceed the expected error bound of the noise-free TCat-concept by more than*

$$4\Delta \tag{4.70}$$

for $C = \frac{1}{\rho R^2}$ *and* R^2 *fixed.*

Proof *Mislabeled documents lie exactly at the margin boundary on the wrong side of the hyperplane given the construction used in the proof of Lemma 2. This implies that their value of* ξ_i *is 2. Following the proof of Theorem 3 and denoting the fraction of noisy examples in the current training set S by* Δ_S *yields*

$$\frac{1}{2}\vec{w}_{sep}^T\vec{w}_{sep} + C(n+1)2\Delta_S \geq \frac{1}{2}\vec{w}_{noise}^T\vec{w}_{noise} + C\sum_{i=1}^{n}\xi_{i,noise} \tag{4.71}$$

Similar to Theorem 3

$$\begin{aligned} Err_{loo}^{n+1}(S) &\leq \frac{\rho R^2}{n+1}\left[\vec{w}_{noise}^T\vec{w}_{noise} + C\sum_{i=1}^{n}\xi_{i,noise}\right] && (4.72)\\ &\leq \frac{\rho R^2}{n+1}\left[\vec{w}_{noise}^T\vec{w}_{noise} + 2C\sum_{i=1}^{n}\xi_{i,noise}\right] && (4.73)\\ &\leq \frac{\rho R^2}{n+1}\left[\vec{w}_{sep}^T\vec{w}_{sep} + 4C(n+1)\Delta_S\right] && (4.74) \end{aligned}$$

Taking the expectation gives an upper bound on the expected error

$$\mathcal{E}(Err^n(h_{SVM})) \leq \frac{\rho R^2}{n+1}\left[\vec{w}_{sep}^T\vec{w}_{sep} + 4C(n+1)\Delta\right] \tag{4.75}$$

which differs from the bound for the noise-free TCat-concept by

$$\rho R^2 4C\Delta \tag{4.76}$$

▮

While the factor of 4 makes this bound somewhat lose, it can easily be tightened and it nevertheless demonstrates that classification noise can be integrated into TCat-concepts.

4.6 Limitations of the Model and Open Questions

Every model abstracts from reality in some sense. While the abstractions used in the model presented here are justified using empirical evidence, it is important to clearly point them out.

First, each document is assumed to exactly follow the same generalized Zipf's law, neglecting variance and discretization inaccuracies that occur especially for short documents. In particular, this implies that all documents are of equal length.

Second, the model fixes the number of occurrences from each word set in the TCat-model. The degree of violation of this assumption is captured in terms of attribute noise. Nevertheless, it might be useful and possible not to specify the exact number of occurrences per word set, but only upper and lower bounds. This could make the model more accurate. However, it comes with the cost of an increased number of parameters, making the model less understandable. While the formal analysis of noise from above demonstrates that the model does not break in the presence of noise, the bounds could be tightened. Along the same lines, parametric noise models could be incorporated to model the types of noise in text-classification problems.

Finally, the general approach taken in this chapter is to model only upper bounds on the error rate. While these are important to derive sufficient conditions for the learnability of text-classification tasks, lower bounds may be of interest as well. They could answer the question of which text-classification tasks cannot be learned with support vector machines.

4.7 Related Work

There is no other model that connects the generalization performance of a learner with the statistical properties of text-classification tasks in a justified way. While some learning algorithms can be analyzed in terms of formal models, these models are inappropriate for text.

The most popular such algorithm is naive Bayes. It is strongly related to probabilistic retrieval models used in information retrieval [Fuhr and Buckley, 1991]. Naive Bayes is commonly justified using assumptions of conditional independence or linked dependence [Cooper, 1991]. For the naive Bayes classifier a sufficient condition for good generalization performance is conditional independence (in terms of a multivariate Bernoulli or multinomial model [McCallum and Nigam, 1998]) of word occurrences given the class label. Unfortunately, this property does not hold for text (e.g. [van Rijsbergen, 1979] [Church, 1995] [Lewis, 1998]). Already when Maron introduced the multivariate Bernoulli model for text, he noted [Maron, 1961, page 410]:

> Clearly this independence assumption is false ...; nevertheless, to facilitate (although degrading) the computations, we can make the independence assumption.

While more complex dependence models can somewhat remove the degree of violation [Sahami, 1998], a principal problem with using generative models for text remains. Finding a generative models for natural language appears much more difficult than solving a text classification task. Therefore, this chapter presented a discriminative model of text classification. It does not model language, but merely constrains the distribution of words enough to ensure good classification accuracy. This way it is possible to avoid unreasonable independence assumptions.

Surprisingly, naive Bayes does provide reasonable classification performance on many tasks despite the violation of independence. A reason might be found based on the ideas in [Domingos and Pazzani, 1997]. They show that sometimes weaker conditions imply good performance of naive Bayes. While this is probably true also for text classification, naive Bayes cannot learn TCat-concepts. It is easy to construct a resonable $\Pr(X)$ so that a naive Bayes classifier, both with the multivariate Bernoulli and the multinomial model, cannot learn the simple $TCat([2:1:4],[1:2:4])$-concept optimally. However, it does find a suboptimal approximation that is still reasonably accurate in many cases. This gives some intuition about the suboptimal, yet surprisingly good performance of naive Bayes on many tasks.

Another model used to describe the properties of text is the 2-Poisson model [Bookstein and Swanson, 1974]. However, like the Bernoulli model it is rejected by tests [Harter, 1975a, Harter, 1975b]. Description oriented approaches [Gövert et al., 1999] [Fuhr and Knorz, 1984] [Fuhr et al., 1991] provide powerful modelling tool and can avoid high-dimensional feature spaces, but require implicit assumptions in the way description vectors are generated.

While different in its motivation and its goal, the work of Papadimitriou et. al is most similar in spirit to the approach presented here [Papadimitriou et al., 1998]. They show that latent semantic indexing leads to a suitable low-dimensional representation, given assumptions about the distribution of words. These assumptions are similar in how they exploit the difference of word distributions. However, they do not show how their assumptions relate to the statistical properties of text and they do not derive generalization-error bounds.

4.8 Summary and Conclusions

This chapter develops the first model of learning text classifiers from examples that makes it possible to connect the statistical properties of text with the generalization performance of the learner. For the first time, it is possible to formalize text classification tasks in a justified way, making them accessible to theoretical analysis. The model is the result of taking a discriminative approach. Unlike conventional generative models, it does not involve unreasonable parametric or independence assumptions. The discriminative model focuses on those properties of the text classification tasks that are sufficient for good generalization performance, avoiding much of the complexity of natural language.

Based on this discriminative model, the chapter proves how support vector machines can achieve good classification performance despite the high-dimensional feature spaces in text classification. This makes SVMs the first learning method with a theoretical justification for its use in text classification. The resulting bounds on the expected generalization error gives a formal understanding of what kind of text-classification task can be solved with support vector machines. This makes it possible to characterize the text classification tasks for which SVMs are appropriate. The chapter identifies that high redundancy, high discriminative power of term sets, and discriminative features in the high-frequency range are sufficient conditions for good generalization.

Beyond text classification, the chapter presents a new bound for the expected error of the support vector machine. Unlike previous bounds of its type, it applies also to biased SVMs and incorporates training error.

While this chapter mainly discusses support vector machines, other learning algorithms can also be analyzed in this model. For example, it should be possible to characterize Boosting in a similar way. Given previous results for Winnow and Perceptron, my conjecture is that maximum-margin methods with respect to L_1-margin will be most effective on tasks with a small set of “strong” features. Such a theoretical understanding of learning algorithms provides the basis for selecting between algorithms based on prior knowledge. It identifies how methods and tasks differ, so that it guides the development of new methods.

Chapter 5

Estimating the Generalization Performance of an SVM Efficiently

Predicting the generalization performance of a learner is one of the central goals of learning theory. The previous chapter approached this question based on an intensional description of the learning task. However, such a model is necessarily coarse, since it operates on a high level of abstraction. Training data can give more details about a learning task than an intensional model with only a few parameters. This chapter explores the problem of predicting the generalization performance of an SVM after training data becomes available.

From a practical perspective, we need accurate and efficient predictions for how well a learner can handle the particular task at hand. Given a particular learning task, a practitioner will ask:

- How well will the learner generalize, given the training examples available?
- Given two parameter settings for the learner, which one leads to better predictions?
- From a set of available hypothesis spaces, which one is best for the task at hand?

The results presented in this section lay the theoretical basis for answering these questions in the context of text classification with SVMs. The aim is to develop operational performance estimators that are of actual use when applying SVMs. This requires that the estimators be both effective and computationally efficient. While the results presented in the following apply to arbitrary learning tasks, special emphasis is put on evaluation measures commonly used in text classification. In particular, the approach is not limited to estimating the error rate. It also covers precision and recall, as well as combined measures like F_1. These measures are far more important for learning useful text classifiers than error rate alone.

This chapter is structured as follows. First, methods for predicting the generalization performance of a learner are reviewed in Section 5.1. While some of these (e.g. uniform convergence bounds) are powerful tools for theory, they are of little use in practical applications. Others (e.g. cross-validation, bootstrap) give good predictions, but are computationally inefficient. Section 5.2 decribes new estimators that exploit special properties of SVMs to overcome these problems, extending results of [Vapnik, 1998, Chapter 10] and [Jaakkola and Haussler, 1999] to general SVMs. The new estimators are both accurate and efficiently computable. After their theoretical justification, the estimators are experimentally tested on three text-classification tasks in Section 5.4. The experiments show that they accurately reflect the actual behavior of SVMs on text-classification tasks.

5.1 Generic Performance Estimators

This section reviews the most common methods for estimating the generalization error

$$Err^n(h_{\mathcal{L}}) = \int L_{0/1}(h_{\mathcal{L}}(\vec{x}), y) d\Pr(\vec{x}, y) = \Pr(h_{\mathcal{L}}(\vec{x}) \neq y | S_n) \tag{5.1}$$

of a learner $\mathcal{L}$ based on a sample S_n of size n, with $L_{0/1}$ being the 0/1-loss function. In particular, these methods are uniform convergence bounds for the training error (Section 5.1.1), Hold-Out Testing (Section 5.1.2), Bootstrapping (Section 5.1.3), and Cross-Validation (Section 5.1.4).

5.1.1 Training Error

The most obvious estimate of the error rate $Err^n(h_{\mathcal{L}})$ is the training error (a.k.a. empirical error, apparent error, resubstitution error)

$$Err^n_{emp}(h_{\mathcal{L}}) = \frac{1}{n} \sum_{i=1}^{n} L_{0/1}(h_{\mathcal{L}}(\vec{x}_i), y_i) \tag{5.2}$$

on the training sample $S = ((\vec{x}_1, y_1), \cdots, (\vec{x}_n, y_n))$. For almost all learners, this estimate is readily available after training. The problem with using $Err^n_{emp}(h_{\mathcal{L}})$ as an estimate for $Err^n(h_{\mathcal{L}})$ is its typically strongly optimistic bias.

$$\mathcal{E}(Err^n_{emp}(h_{\mathcal{L}})) << \mathcal{E}(Err^n(h_{\mathcal{L}})) \tag{5.3}$$

Especially for learners that minimize training error, $Err^n_{emp}(h_{\mathcal{L}})$ is usually much lower than the true error $Err^n(h_{\mathcal{L}})$, since it is measured on the same data which the learner used to find the hypothesis. For SVMs this effect is most extreme when kernels of high capacity are used. They fit the data perfectly and so have a training error of zero, while the true error rate can be high.

VC-theory identifies the situations for which $Err^n_{emp}(h_{\mathcal{L}})$ is sufficiently close to $Err^n(h_{\mathcal{L}})$ and places an upper bound on the difference. The bound depends only on the VC-dimension $d_{\mathcal{H}}$ of the hypothesis space $\mathcal{H}$ that the learner considers and the number of training examples n, but it is independent of the learner. One bound on the difference $|Err^n(h_{\mathcal{L}}) - Err^n_{emp}(h_{\mathcal{L}})|$ can be derived from the bound in [Wapnik and Tscherwonenkis, 1979], page 161:

$$\Pr(|Err^n(h_{\mathcal{L}}) - Err^n_{emp}(h_{\mathcal{L}})| > \epsilon) \leq 6 \left(\frac{e\,n}{d_{\mathcal{H}}}\right)^{d_{\mathcal{H}}} exp\left(-\frac{\epsilon^2(n-1)}{4}\right) \tag{5.4}$$

Using this bound can help quantify, the amount by which $Err^n_{emp}(h_{\mathcal{L}})$ underestimates the true error. This leads to an upper bound on the error rate, as is desirable in practice. Solving (5.4) for $Err^n(h_{\mathcal{L}})$ gives such an upper bound. With probability $1 - \eta$

$$Err^n(h_{\mathcal{L}}) \leq Err^n_{emp}(h_{\mathcal{L}}) + 2\sqrt{\frac{d_{\mathcal{H}}(ln\frac{2\,n}{d_{\mathcal{H}}} + 1) - ln\frac{\eta}{4}}{n}} \tag{5.5}$$

Unfortunately, for most practical applications this bound is of little use. For the amount of data usually available, it is too loose to make reasonable predictions about $Err^n(h_\mathcal{L})$. To a large extent this is due to the fact that the bound is independent of the learning task $\Pr(\vec{x}, y)$. While this remarkable property makes the bound a very universal tool, it is "worst-case" with respect to all $\Pr(\vec{x}, y)$. The methods discussed in the following use the training sample as an approximation to $\Pr(\vec{x}, y)$.

Let us finally look at the expected difference between training error and true error for $n > d_\mathcal{H}$. Using a result from [Devroye et al., 1996, page 208], it is easy to translate bound (5.4) into:

$$\mathcal{E}(|Err^n(h_\mathcal{L}) - Err^n_{emp}(h_\mathcal{L})|^2) \leq O\left(\frac{d_\mathcal{H}\, ln(\frac{n}{d_\mathcal{H}})}{n}\right) \tag{5.6}$$

5.1.2 Hold-Out Testing

In hold-out testing (see e.g. [Devroye et al., 1996]) the sample S_n is divided randomly into two parts $S_l^{train} \cup S_k^{val} = S_n$ of size l and k. The learner uses the training sample S_l^{train} for training, while the validation sample S_k^{val} serves as an independent test set for estimating the true error of the classification rule. The hold-out estimate $Err_{ho}^{l,k}(h_\mathcal{L})$ is as follows:

$$Err_{ho}^{l,k}(h_\mathcal{L}) = \frac{1}{k} \sum_{(\vec{x}_i, y_i) \in S_k^{val}} L_{0/1}(h_\mathcal{L}(\vec{x}_i), y_i) \tag{5.7}$$

While $Err_{ho}^{l,k}(h_\mathcal{L})$ is an unbiased estimate of $Err^l(h_\mathcal{L})$, it is not unbiased with respect to $Err^n(h_\mathcal{L})$. The learner is trained with only l training examples instead of the full sample S_n containing n examples. Since we typically expect the performance of the learner to increase with more training data, the hold-out estimate $Err_{ho}^{l,k}(h_\mathcal{L})$ is negatively biased.

$$\mathcal{E}(Err_{ho}^{l,k}(h_\mathcal{L})) > \mathcal{E}(Err^n(h_\mathcal{L})) \tag{5.8}$$

The expected deviation $\mathcal{E}((Err^l(h_\mathcal{L}) - Err_{ho}^{l,k}(h_\mathcal{L}))^2)$ can easily be calculated, since each test on an example from the validation set is an independent Bernoulli trial.

$$\mathcal{E}(|Err^l(h_\mathcal{L}) - Err_{ho}^{l,k}(h_\mathcal{L})|^2) \leq \frac{1}{4\,k} \tag{5.9}$$

Equation (5.8) and (5.9) exhibit a trade-off in selecting l and k. The larger l, the smaller the bias. At the same time, the variance increases with decreasing $k = n - l$. The optimal choice of l and k depend on the learner $\mathcal{L}$, the hypothesis space $\mathcal{H}$, and the learning task $\Pr(\vec{x}, y)$ [Kearns, 1996]. Nevertheless, there are good heuristics for selecting reasonable values for l and k [Kearns, 1996].

Let us finally also look at worst-case bounds for the deviation. Using Hoeffding bounds [Hoeffding, 1963] it holds that

$$\Pr(|Err^l(h_\mathcal{L}) - Err_{ho}^{l,k}(h_\mathcal{L})| > \epsilon) \leq 2\,exp(-2\,k\,\epsilon^2) \tag{5.10}$$

Experimental results for hold-out estimates are given in [Kearns et al., 1997].

The hold-out estimate is efficiently computable. It involves one training run on l training examples and the classification of k test examples.

5.1.3 Bootstrap and Jackknife

The methods based on bootstrap [Efron, 1983][Efron, 1982][Shao and Tu, 1995] or jackknife [Efron, 1982] statistics aim to estimate the bias of the training error $Err^n_{emp}(h_{\mathcal{L}})$. All bootstrap methods make use of bootstrap samples $S^b_m = ((\vec{x}^b_1, y^b_1), \cdots, (\vec{x}^b_m, y^b_m))$, each generated by independently drawing m examples from the training sample S_n with replacement. Usually, the size of the bootstrap samples is chosen to be the same as the size of the training sample (i.e. $m = n$). The learner is trained on each bootstrap sample and outputs a corresponding hypothesis $h^i_{\mathcal{L}}$. For the k-th bootstrap sample the estimate of the bias is

$$bias^n_k = \sum_{i=1}^{n} \left(\frac{1}{n} - \frac{1}{n} \sum_{j=1}^{m} L_{0/1}(\vec{x}^b_j, \vec{x}_i) \right) L_{0/1}(h^i_{\mathcal{L}}(\vec{x}_i), y_i) \tag{5.11}$$

The individual estimates $bias^n_k$ are averaged over B bootstrap samples to remove the randomness introduced by the sampling.

$$bias^n = \sum_{k=1}^{B} bias^n_k \tag{5.12}$$

The bootstrap estimate $Err^n_b(h_{\mathcal{L}})$ of the true error $Err^n(h_{\mathcal{L}})$ is the training error $Err^n_{emp}(h_{\mathcal{L}})$ minus the bootstrap estimate $bias^n$ of the bias.

$$Err^n_{boot}(h_{\mathcal{L}}) = Err^n_{emp}(h_{\mathcal{L}}) - bias^n \tag{5.13}$$

A jackknife approximation to the bootstrap estimate is described in [Efron, 1982]. Other version of the bootstrap estimator can be found in [Efron, 1983] and [Efron and Tibshirani, 1993].

Little is known about the bias and the variance of the bootstrap estimate $R^n_b(h_{\mathcal{L}})$. Experimental evidence suggests that its bias is comparatively large, while the variance is small [Breiman et al., 1984][Bailey and Elkan, 1993][Kohavi, 1995]. Davison and Hall [Davison and Hall, 1992] provide some theoretical results for a particular small example that supports this observation.

The computational costs for computing the bootstrap estimate are high. The learner is invoked B times, once for each bootstrap sample. In addition, the training set of n examples needs to be classified each time. Typical values for B are between 10 and 100.

5.1.4 Cross-Validation and Leave-One-Out

The most popular method for estimating the generalization error of a classification rule is cross-validation (a.k.a. delete-d method, rotation estimate) [Lunts and Brailovskiy, 1967][Stone, 1974][Lachenbruch and Mickey, 1968]. While there are several versions of the cross-validation estimator, most theoretical results concern the leave-one-out estimator described in the following. From the training sample $S = ((\vec{x}_1, y_1), \cdots, (\vec{x}_n, y_n))$ the first example $(\vec{x}_1, y_1)$ is removed. The resulting sample $S^{\backslash 1} = ((\vec{x}_2, y_2), \cdots, (\vec{x}_n, y_n))$ is used for training, leading to a classification rule $h^{\backslash 1}_{\mathcal{L}}$. This classification rule is tested on the held-out example $(\vec{x}_1, y_1)$. This process is repeated for all training examples. The number of misclassifications divided by n is the leave-one-out estimate of the generalization error.

$$Err^n_{loo}(h_{\mathcal{L}}) = \frac{1}{n} \sum_{i=1}^{n} L_{0/1}(h^{\backslash i}_{\mathcal{L}}(\vec{x}_i), y_i) \tag{5.14}$$

Lunts and Brailovskiy showed that this estimate is almost unbiased in the following sense [Lunts and Brailovskiy, 1967].

Theorem 5 ([Lunts and Brailovskiy, 1967] Bias of Leave-One-Out Estimator) *The leave-one-out estimator is almost unbiased; that is*

$$\mathcal{E}(R_{loo}^{n}(h_{\mathcal{L}})) = \mathcal{E}(R^{n-1}(h_{\mathcal{L}})) \tag{5.15}$$

The expectation on the left-hand side is over training sets of size n, the one on the right-hand side is over training sets of size $n-1$.

Proof *Abbreviating Z_i for $(\vec{x}_i, y_i)$, the theorem follows from the following chain of transformations for all bounded loss functions:*

$$\mathcal{E}(R_{loo}^{n}(h_{\mathcal{L}})) = \int \frac{1}{n}\sum_{i=1}^{n} L(h_{\mathcal{L}}^{\setminus i}(\vec{x}_i), y_i) d\Pr(Z_1)...d\Pr(Z_n) \tag{5.16}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \int L(h_{\mathcal{L}}^{\setminus i}(\vec{x}_i), y_i) d\Pr(Z_1)...d\Pr(Z_n) \tag{5.17}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \int \left[\int L(h_{\mathcal{L}}^{\setminus i}(\vec{x}_i), y_i) d\Pr(Z_i)\right] d\Pr(Z_1)...d\Pr(Z_{i-1})d\Pr(Z_{i+1})...d\Pr(Z_n) \tag{5.18}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \int R^{n-1}(h_{\mathcal{L}}) d\Pr(Z_1)...d\Pr(Z_{i-1})d\Pr(Z_{i+1})...d\Pr(Z_n) \tag{5.19}$$

$$= \frac{1}{n} n \int R^{n-1}(h_{\mathcal{L}}) d\Pr(Z_1)...d\Pr(Z_{n-1}) \tag{5.20}$$

$$= \mathcal{E}(R^{n-1}(h_{\mathcal{L}})) \tag{5.21}$$

∎

The theorem identifies that the bias depends on how much a single training example changes the performance of the learner. On most practical problems this is negligibly small.

The variability of the leave-one-out estimator depends on both the learning algorithms as well as the learning task. The dependence is captured in the following bound [Rogers and Wagner, 1978][Devroye and Wagner, 1976]. The bound holds for all classifiers that are independent of the ordering of the training examples.

Theorem 6 ([Rogers and Wagner, 1978][Devroye and Wagner, 1976] Variability of the Leave-One-Out Estimator) *It holds that*

$$\mathcal{E}(|Err_{loo}^{n}(h_{\mathcal{L}}) - Err^{n}(h_{\mathcal{L}})|^2) \leq \frac{1}{n} + 6\ \Pr(h_{\mathcal{L}}^{n}(\vec{x}) \neq h_{\mathcal{L}}^{n-1}(\vec{x})) \tag{5.22}$$

$\Pr(h_{\mathcal{L}}^{n}(\vec{x}) \neq h_{\mathcal{L}}^{n-1}(\vec{x}))$ is the probability that a classification rule $h_{\mathcal{L}}^{n}(\vec{x})$ (trained using all n examples) will disagree on a randomly drawn example with a classification rule $h_{\mathcal{L}}^{n-1}(\vec{x}))$ (trained on the same sample with one example removed). The proof of the theorem is given in [Devroye et al., 1996, pages 411-413]. Theorem 6 can be used to upper bound the variability of the leave-one-out estimator for the special case of local classification rules [Devroye and Wagner, 1979b][Devroye and Wagner, 1979a]. Somewhat surprising results about the asymptotic behavior of the leave-one-out estimate in the general case are given in [Stone, 1977]. Shao and Tu give a summary of the discussion about using cross-validation for model selection in linear models [Shao and Tu, 1995]. A similar bound on the variability of the leave-one-out estimator is given in [Lunts and Brailovskiy, 1967]. A more general bound [Kearns and Ron, 1997] based on uniform convergence arguments is discussed in Section 5.2.1.

The computational demands of the leave-one-out estimator are high. The learner is invoked n times on training sets of $n-1$ examples. This is prohibitively expensive for all but small n. To reduce running time it is common practice to combine cross-validation with hold-out testing [Toussaint and Donaldson, 1970][Mitchell, 1997]. Instead of training on $n-1$ examples and testing on only one, the training set is partitioned into k folds. Assuming that $\frac{n}{k}$ is an integer, each fold contains $\frac{n}{k}$ examples. The learner now repeatedly trains on $k-1$ folds and each resulting classification rule is tested on the remaining fold. The average performance is the k-fold cross-validation estimate $Err^n_{kcv}(h_{\mathcal{L}})$ (a.k.a. delete-$(\frac{n}{k})$ estimate, rotation estimate). Note that k-fold cross-validation has a larger bias than leave-one-out.

$$\mathcal{E}(Err^n_{kcv}(h_{\mathcal{L}})) = \mathcal{E}(Err^{n-\frac{n}{k}}(h_{\mathcal{L}})) \tag{5.23}$$

Experimental results show that cross-validation is a good estimator of the generalization performance. It is repeatedly reported to have lower bias than the bootstrap estimate [Efron, 1983][Breiman et al., 1984][Kohavi, 1995][Bailey and Elkan, 1993], but typically has higher variability. The variability of 10-fold cross-validation tends to be lower than that of leave-one-out [Kohavi, 1995][Bailey and Elkan, 1993][Efron, 1983][Devroye et al., 1996].

5.2 $\xi\alpha$-Estimators

While the estimation methods in the previous section are applicable to arbitrary learning algorithms, this section develops special estimators for support vector machines. The estimators proposed in the following are based on the leave-one-out method, but require an order of magnitude less computation time due to particular properties of the SVM. In particular, they do not require actually performing resampling and retraining, but can be applied directly after training the learner. The inputs to the estimators are the vector $\vec{\alpha}$ solving the dual SVM training problem (Optimization Problem 4) and the vector $\vec{\xi}$ from the solution of the primal SVM training problem(Optimization Problem 3). Due to this dependence, they will be called $\xi\alpha$-estimators in the following.

As already argued above, in text classification error rate alone is not necessarily a good performance measure. Instead, scores based on precision and recall are of widespread use. To get useful tools for text classification, I will propose and explore $\xi\alpha$-estimators not only for the error rate (Section 5.2.1), but also for the recall (Section 5.2.2), the precision (Section 5.2.3), and the $F1$-measure (Section 5.2.4).

5.2.1 Error Rate

This section starts with the definition of the $\xi\alpha$-estimator of the error rate. Based on the solution $\vec{\alpha}$ of the dual SVM training problem and the vector of training losses $\vec{\xi}$, the $\xi\alpha$-estimator of the error rate $Err^n_{\xi\alpha}(h_{\mathcal{L}})$ is defined as follows.

Definition 9 ($\xi\alpha$-Estimator of the Error Rate) *For stable soft-margin SVMs, the $\xi\alpha$-estimator of the error rate is*

$$Err^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{d}{n} \quad \textit{with} \quad d = |\{i : (\rho\alpha_i R^2_{\Delta} + \xi_i) \geq 1\}| \tag{5.24}$$

with ρ equals 2. $\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 4 and 3 (or 6 and 5) on the training set S_n. R^2_{Δ} is an upper bound on $c \leq \mathcal{K}(\vec{x}, \vec{x}') \leq c + R^2_{\Delta}$ for all $\vec{x}, \vec{x}'$ and some constant c.

The definition introduces the parameter ρ. While the theoretical results that are derived below assume $\rho = 2$, we will see that $\rho = 1$ is a good choice for text classification[1]. The key quantity in definition 9 is d. d counts the number of training examples for which the inequality $(\rho\alpha_i R_\Delta^2 + \xi_i) \geq 1$ holds. But how does one come to this definition of d and what exactly does d count?

The key idea to the $\xi\alpha$-estimator of the error rate is a connection between the training examples for which the inequality $(\rho\alpha_i R_\Delta^2 + \xi_i) \geq 1$ holds and those training examples that can produce an error in leave-one-out testing. In particular, if an example $(\vec{x}_i, y_i)$ is classified incorrectly by a SVM trained on the subsample $S_n^{\backslash i}$, then example $(\vec{x}_i, y_i)$ *must* fulfill the inequality $(\rho\alpha_i R_\Delta^2 + \xi_i) \geq 1$ for a SVM trained on the full sample S_n. This implies that d is an upper bound on the number of leave-one-out errors. The following lemma establish this result formally.

Lemma 4 (Bound on Leave-One-Out Error of Stable Soft-Margin SVMs) *The number of leave-one-out errors $\sum_{i=1}^n L_{0/1}(h_\mathcal{L}^{\backslash i}(\vec{x}_i), y_i)$ of stable soft-margin SVMs on a training set S_n is bounded by*

$$\sum_{i=1}^n L_{0/1}(h_\mathcal{L}^{\backslash i}(\vec{x}_i), y_i) \leq |\{i : (2\,\alpha_i R^2 + \xi_i) \geq 1\}| \tag{5.25}$$

$\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 4 and 3 (or 6 and 5) on the training set S_n. R^2 is an upper bound on $\mathcal{K}(\vec{x}, \vec{x})$ and $\mathcal{K}(\vec{x}, \vec{x}') \geq 0$.

Proof *An error on a left out example $(\vec{x}_t, y_t)$ occurs when at the solution of*

$$W_t(\vec{\alpha}^t) = max_{\vec{0} \leq \vec{\alpha}^t \leq \vec{C}} 1^T \vec{\alpha}^t - \frac{1}{2} \vec{\alpha}^{tT} Q^t \vec{\alpha}^t \wedge \vec{y}^{tT} \vec{\alpha}^t = 0 \tag{5.26}$$

where $\vec{\alpha}^t$, $\vec{y}^t$, and Q^t have the t-th example removed, the expression

$$y_t \left[\sum_{i \neq t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b^t \right] > 0 \tag{5.27}$$

is false. What follows in this proof are conditions for when this expression must be true based on the soft-margin SVM solution

$$W(\vec{\alpha}) = max_{\vec{0} \leq \vec{\alpha} \leq \vec{C}} 1^T \vec{\alpha} - \frac{1}{2} \vec{\alpha}^T Q \vec{\alpha} \wedge \vec{y}^T \vec{\alpha} = 0 \tag{5.28}$$

that involves all n training examples. Three cases can occur based on the optimal value of α_t:

Case $\alpha_t = 0$: *Example $(\vec{x}_t, y_t)$ is not a support vector. Then $W_t(\vec{\alpha}^t) = W(\vec{\alpha})$ and $y_t \sum_{i \neq t} y_i \alpha_i^t \mathcal{K}(\vec{x}_t, \vec{x}_i) = y_t \sum_{i \neq t} y_i \alpha_i \mathcal{K}(\vec{x}_t, \vec{x}_i)$. Since the t-th example is not a support vector, we know that $y_t \sum_{i \neq t} y_i \alpha_i \mathcal{K}(\vec{x}_t, \vec{x}_i) \geq 1$ and so (5.27) must be true. So the t-th example cannot produce a leave-one-out error. Finally, it is not counted as a leave-one-out error, since $\alpha_i + \xi_i = 0$ for non-support vectors.*

Case $0 < \alpha_t < C$: *Example $(\vec{x}_t, y_t)$ is a support vector. From the solution $\vec{\alpha}^t$ of $W_t(.)$, the following construction produces a feasible point $\vec{\beta}$ for $W(.)$.*

$$\beta_i = \begin{cases} \alpha_i^t & if \;\; \alpha_i^t = 0 \vee \alpha_i^t = C \\ \alpha_i^t - y_i y_t \nu_i & if \;\; i \in SV^t \\ \alpha_i & if \;\; i = t \end{cases} \tag{5.29}$$

[1] Although it is not proven here, it is straightforward to verify that all theoretical results presented in the following also hold for unbiased soft-margin SVMs with $\rho = 1$.

ν_i has to fulfill the following constraints. Let SV^t be the set of indices corresponding to support vectors of the solution $W_t(\vec{\alpha}^t)$ that are not at the upper bound C (that is $0 < \alpha_i^t < C$). Then $\nu_i = 0$ for all $i \notin SV^t$. For $i \in SV^t$ the ν_i are chosen to be non-negative and so that $\sum_{i \in SV^t} \nu_i = \alpha_t$ and $0 \leq \beta_i \leq C$. Finding such ν_i is always possible, if there are at least two support vectors not at the upper bound C. The existence of such two vectors follows from the assumption that the SVMs solution is stable. From the construction of the ν_i it follows that $\vec{y}^T \vec{\beta} = 0$ and $0 \leq \beta_i \leq C$. So $\vec{\beta}$ is a feasible point of $W(.)$. After a series of transformations, $W(\vec{\beta})$ can be written as

$$\begin{aligned} W(\vec{\beta}) &= W_t(\vec{\alpha}^t) - \frac{1}{2}\alpha_t^2 \mathcal{K}(\vec{x}_t, \vec{x}_t) + \alpha_t & (5.30)\\ &-\alpha_t y_t \sum_{i \neq t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) - y_t \left[\sum_{i \in SV^t} \nu_i \left(y_i - \sum_{j \neq t} \alpha_j^t y_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \right) \right] & (5.31)\\ &-\frac{1}{2} \sum_{i \in SV^t} \sum_{j \in SV^t} \nu_i \nu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \alpha_t \sum_{i \in SV^t} \nu_i \mathcal{K}(\vec{x}_i, \vec{x}_t) & (5.32) \end{aligned}$$

For support vectors not at the upper bound the expression in the round brackets (line 5.31) equals the threshold b^t of the classification rule (compare equation (3.21)). Exploiting also that $\sum_{i \in SV^t} \nu_i = \alpha_t$ by construction, it is possible to write:

$$\begin{aligned} \alpha_t y_t \left[\sum_{i \neq t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b^t \right] &= -W(\vec{\beta}) + W_t(\vec{\alpha}^t) - \frac{1}{2}\alpha_t^2 \mathcal{K}(\vec{x}_t, \vec{x}_t) + \alpha_t & (5.33)\\ &-\frac{1}{2} \sum_{i \in SV^t} \sum_{j \in SV^t} \nu_i \nu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \alpha_t \sum_{i \in SV^t} \nu_i \mathcal{K}(\vec{x}_i, \vec{x}_t) & (5.34) \end{aligned}$$

Let us now do a similar construction for producing a feasible point $\vec{\gamma}$ of $W_t(.)$ based on the solution $\vec{\alpha}$ of $W(.)$.

$$\gamma_i = \begin{cases} \alpha_i & if \quad \alpha_i = 0 \vee \alpha_i = C \\ \alpha_i + y_i y_t \mu_i & if \quad i \in SV^{\setminus t} \end{cases} \tag{5.35}$$

SV is the set of indices corresponding to support vectors not at the upper bound for the solution $\vec{\alpha}$ (that is $0 < \alpha_i < C$). $SV^{\setminus t}$ excludes the index t corresponding to the left-out example. μ_i is chosen non-negative for all $i \in SV^{\setminus t}$ such that $\sum_{i \in SV^{\setminus t}} \mu_i = \alpha_t$ and $0 \leq \gamma_i \leq C$. From the construction of the μ_i it follows that $\vec{y}^T \vec{\gamma} = 0$ and so $\vec{\gamma}$ is a feasible point of $W_t(.)$. After a series of transformations, $W_t(\vec{\gamma})$ can be written as

$$\begin{aligned} W^t(\vec{\gamma}) &= W(\vec{\alpha}) + \frac{1}{2}\alpha_t^2 \mathcal{K}(\vec{x}_t, \vec{x}_t) - \alpha_t & (5.36)\\ &+\alpha_t y_t \sum_{i \neq t} \alpha_i y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + y_t \left[\sum_{i \in SV^{\setminus t}} \mu_i \left(y_i - \sum_{j \neq t} \alpha_j y_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \right) \right] & (5.37)\\ &-\frac{1}{2} \sum_{i \in SV^{\setminus t}} \sum_{j \in SV^{\setminus t}} \mu_i \mu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) & (5.38) \end{aligned}$$

Now, the expression in the round parentheses equals the threshold b of the classification rule based on all examples plus $\alpha_t y_t \mathcal{K}(\vec{x}_i, \vec{x}_t)$. Substituting and rearraging leads to the equation

$$-W(\vec{\alpha}) = -W^t(\vec{\gamma}) + \frac{1}{2}\alpha_t^2 \mathcal{K}(\vec{x}_t, \vec{x}_t) - \alpha_t \tag{5.39}$$

$$+\alpha_t y_t \left[\sum_{i\neq t} \alpha_i y_i \mathcal{K}(\vec{x}_t,\vec{x}_i)+b\right] \tag{5.40}$$

$$-\frac{1}{2}\sum_{i\in SV^{\setminus t}}\sum_{j\in SV^{\setminus t}} \mu_i\mu_j\mathcal{K}(\vec{x}_i,\vec{x}_j)+\alpha_t\sum_{i\in SV^{\setminus t}}\mu_i\mathcal{K}(\vec{x}_i,\vec{x}_t) \tag{5.41}$$

It is now possible to substitute $W(\vec{\alpha})$ for $W(\vec{\beta})$ in equation (5.33). Since $W(\vec{\alpha})$ is larger than $W(\vec{\beta})$ by definition, this results in the inequality

$$\alpha_t y_t\left[\sum_{i\neq t}\alpha_i^t y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b^t\right] \;\geq\; W_t(\vec{\alpha}^t)-W^t(\vec{\gamma}) \tag{5.42}$$

$$+\alpha_t y_t\left[\sum_{i\neq t}\alpha_i y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b\right] \tag{5.43}$$

$$-\frac{1}{2}\sum_{i\in SV^{\setminus t}}\sum_{j\in SV^{\setminus t}}\mu_i\mu_j\mathcal{K}(\vec{x}_i,\vec{x}_j)+\alpha_t\sum_{i\in SV^{\setminus t}}\mu_i\mathcal{K}(\vec{x}_i,\vec{x}_t) \tag{5.44}$$

$$-\frac{1}{2}\sum_{i\in SV^t}\sum_{j\in SV^t}\nu_i\nu_j\mathcal{K}(\vec{x}_i,\vec{x}_j)+\alpha_t\sum_{i\in SV^t}\nu_i\mathcal{K}(\vec{x}_i,\vec{x}_t) \tag{5.45}$$

Similarly, $W_t(\vec{\alpha}^t)\geq W^t(\vec{\gamma})$ so that

$$\alpha_t y_t\left[\sum_{i\neq t}\alpha_i^t y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b^t\right] \;\geq\; \alpha_t y_t\left[\sum_{i\neq t}\alpha_i y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b\right] \tag{5.46}$$

$$-\frac{1}{2}\sum_{i\in SV^{\setminus t}}\sum_{j\in SV^{\setminus t}}\mu_i\mu_j\mathcal{K}(\vec{x}_i,\vec{x}_j)+\alpha_t\sum_{i\in SV^{\setminus t}}\mu_i\mathcal{K}(\vec{x}_i,\vec{x}_t) \tag{5.47}$$

$$-\frac{1}{2}\sum_{i\in SV^t}\sum_{j\in SV^t}\nu_i\nu_j\mathcal{K}(\vec{x}_i,\vec{x}_j)+\alpha_t\sum_{i\in SV^t}\nu_i\mathcal{K}(\vec{x}_i,\vec{x}_t) \tag{5.48}$$

The term $\alpha_t\sum_{i\in SV^t}\nu_i\mathcal{K}(\vec{x}_i,\vec{x}_t)$ is non-negative, since all ν_i and $\mathcal{K}(\vec{x}_i,\vec{x}_t)$ are non-negative. The same holds for $\alpha_t\sum_{i\in SV^{\setminus t}}\mu_i\mathcal{K}(\vec{x}_i,\vec{x}_t)$. Furthermore, $\frac{1}{2}\sum_{i\in SV^{\setminus t}}\sum_{j\in SV^{\setminus t}}\mu_i\mu_j\mathcal{K}(\vec{x}_i,\vec{x}_j)$ $\leq\frac{1}{2}\alpha_t^2R^2$ and $\frac{1}{2}\sum_{i\in SV^t}\sum_{j\in SV^t}\nu_i\nu_j\mathcal{K}(\vec{x}_i,\vec{x}_j)\leq\frac{1}{2}\alpha_t^2R^2$, since the $K(\vec{x}_i,\vec{x}_j)$ form a positive semi-definite matrix with the diagonal elements bounded from above by R^2. For a positive semi-definite matrix, the off-diagonal elements must be less than or equal to R^2. Using these inequalities and dividing by α_t, it is possible to write

$$y_t\left[\sum_{i\neq t}\alpha_i^t y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b^t\right] \;\;\geq\;\; y_t\left[\sum_{i\neq t}\alpha_i y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b\right]-\alpha_tR^2 \tag{5.49}$$

This means a leave-one-out error can occur only when

$$y_t\left[\sum_{i\neq t}\alpha_i y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b\right]-\alpha_tR^2\leq 0 \tag{5.50}$$

Or equivalently, after adding $y_t\alpha_t y_t\mathcal{K}(\vec{x_t},\vec{x_t})$ and exploiting the fact that for support vectors $y_t\left[\sum_{i=1}^n\alpha_i y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b\right]=1$

$$1 \;\;\leq\;\; \alpha_t\mathcal{K}(\vec{x}_t,\vec{x}_t)+\alpha_tR^2 \tag{5.51}$$

$$\Rightarrow 1 \;\;\leq\;\; 2\alpha_tR^2 \tag{5.52}$$

Since $\xi_i = 0$ for support vectors, the condition $(2\alpha_t R^2 + \xi_t) \geq 1$ is always fulfilled if $(\vec{x}_t, y_t)$ produces a leave-one-out error.

Case $\alpha_t = C$: *Example $(\vec{x}_t, y_t)$ is a bounded support vector. The argumentation follows that in the case of regular support vectors up to the point where it is shown that a leave-one-out error can occur only, if*

$$y_t \left[\sum_{i \neq t} \alpha_i y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b \right] - \alpha_t R^2 \leq 0 \tag{5.53}$$

Adding $y_t \alpha_t y_t \mathcal{K}(\vec{x}_t, \vec{x}_t)$ and exploiting the fact that $y_t \left[\sum_{i=1}^n \alpha_i y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b\right] = 1 - \xi_t$ for bounded support vectors

$$\begin{aligned} 1 - \xi_t &\leq \alpha_t \mathcal{K}(\vec{x}_t, \vec{x}_t) + \alpha_t R^2 && (5.54) \\ \Rightarrow 1 &\leq 2\alpha_t R^2 + \xi_t && (5.55) \end{aligned}$$

This shows that also in the case of a bounded support vector the condition $(2\alpha_t R^2 + \xi_t) \geq 1$ is always fulfilled if $(\vec{x}_t, y_t)$ produces a leave-one-out error. ▮

The idea of connecting the leave-one-out error with properties of the solution vector $\vec{\alpha}$ goes back to Vapnik ([Vapnik, 1998], pages 418-421). Unlike the work presented here, Vapnik's result is limited in three ways: (a) the training data must be separable, (b) it only holds for the special case of hyperplanes passing through the origin, and (c) it is used to derive bounds on the expected error, not estimators. Jaakola and Haussler [Jaakkola and Haussler, 1999] present a generalized bound for inseparable data that is similar to that of Lemma 4. Nevertheless, like Vapnik's bound it is restricted to hyperplanes passing through the origin and does not apply to regular SVMs. Approximations to the leave-one-out error of SVMs without guaranteeing an upper bound were recently proposed in [Wahba, 1999] and [Opper and Winther, 2000]. An additional difference is that their approach requires computing the inverse for part of the Hessian, making it computationally more expensive.

While Lemma 4 is valid for all kernel functions that return positive values, it is tightest when the minimum value is zero. The following lemma shows that this can always be achieved.

Lemma 5 (Invariance of Soft-Margin SVM) *The soft-margin SVM is invariant under addition of a real value c to the kernel function.*

Proof *Let $\mathcal{K}'$ be a kernel function derived from a positive semi-definite kernel $\mathcal{K}$ by adding a constant $c \in \Re$ to $\mathcal{K}$. Then the soft-margin SVM solution involving $\mathcal{K}$ is also a solution to the soft-margin SVM problem involving $\mathcal{K}'$. The following series of transformations shows this. Subject to $\sum_{i=1}^n y_i \alpha_i = 0$ it holds for all $\vec{\alpha}$ that*

$$\begin{aligned} W(\vec{\alpha}) &= -\sum_{i=1}^n \alpha_i + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j (\mathcal{K}(\vec{x}_i, \vec{x}_j) + c) && (5.56) \\ &= -\sum_{i=1}^n \alpha_i + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \frac{c}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j && (5.57) \\ &= -\sum_{i=1}^n \alpha_i + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \frac{c}{2} \sum_{i=1}^n y_i \alpha_i \sum_{j=1}^n y_j \alpha_j && (5.58) \\ &= -\sum_{i=1}^n \alpha_i + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \mathcal{K}(\vec{x}_i, \vec{x}_j) && (5.59) \end{aligned}$$

Similarly, the resulting classification rules can be shown to be equivalent. For the solution vector $\vec{\alpha}$

$$\vec{w}\cdot\vec{x} = \sum_{i=1}^{n} \alpha_i y_i (\mathcal{K}(\vec{x}_i, \vec{x}_j) + c) \tag{5.60}$$

$$= \sum_{i=1}^{n} \alpha_i y_i \mathcal{K}(\vec{x}_i, \vec{x}_j) + c \sum_{i=1}^{n} \alpha_i y_i \tag{5.61}$$

$$= \sum_{i=1}^{n} \alpha_i y_i \mathcal{K}(\vec{x}_i, \vec{x}_j) \tag{5.62}$$

∎

The previous two lemmas and Theorem 5 complete the tools needed to characterize the bias of the $\xi\alpha$-estimator of the error rate.

Theorem 7 (Bias of $\xi\alpha$-Estimator of the Error Rate) *The $\xi\alpha$-estimator of the error rate is conservatively biased in the following sense*

$$\mathcal{E}(Err^n_{\xi\alpha}(h_\mathcal{L})) \geq \mathcal{E}(Err^{n-1}(h_\mathcal{L})) \tag{5.63}$$

Proof *Theorem 5 shows that the leave-one-out estimator $Err^n_{loo}(h_\mathcal{L})$ of the error rate on training sets of size n gives an unbiased estimate of the error rate after training on $n-1$ examples. After adding a constant c to the kernel function so that $min\mathcal{K}(\vec{x}_i, \vec{x}) = 0$, the theorem follows directly from the bound in Lemma 4 using Lemma 5. The lemma establishes that $Err^n_{\xi\alpha}(h_\mathcal{L}) \geq Err^n_{loo}(h_\mathcal{L})$ with $R^2 = c + max\mathcal{K}(\vec{x}, \vec{x})$ and therefore $\mathcal{E}(Err^n_{\xi\alpha}(h_\mathcal{L})) \geq \mathcal{E}(Err^{n-1}(h_\mathcal{L}))$.* ∎

In other words, the theorem states that the $\xi\alpha$-estimator tends to overestimate the true error rate. This means that "on average" the estimate is higher than the true error. Given a low variance of the estimate, it is now possible to guarantee with a certain probability that the true error is lower than the estimate. Nevertheless, known bounds on the variance depend on further assumptions about the learner and/or the learning task. Theorem 6 requires that the probability $\Pr(h^n_\mathcal{L}(\vec{x}) \neq h^{n-1}_\mathcal{L}(\vec{x}))$ is small. For SVMs this quantity depends on the learning task. Bounds on the variability of the leave-one-out estimator presented in [Kearns and Ron, 1997] are independent of the learning task. Assuming that the learner returns a classification rule with minimum training error, Kearns and Ron bound the variability based on the VC-dimension of the hypothesis space.

Theorem 8 ([Kearns and Ron, 1997] Bound on the Variability of $Err^n_{loo}(h_\mathcal{L})$) *Let A be any algorithm performing training error minimization over a hypothesis space h of VC-dimension d. Then for every $\nu > 0$, with probability $1 - \nu$,*

$$|Err^n_{loo}(h_\mathcal{L}) - Err^n(h_\mathcal{L})| \leq \frac{8\sqrt{\frac{(d+1)(ln(9\,n/d)+2)}{n}}}{\nu} \tag{5.64}$$

This bound can easily be used to upper-bound the probability that the $\xi\alpha$-estimator underestimates the true error. Nevertheless, the bound would be too loose to be of practical importance. Therefore, the variability of the $\xi\alpha$-estimator will be assessed empirically in Section 5.4.

5.2.2 Recall

The presentation of the $\xi\alpha$-estimator of the recall follows the structure of the previous section. First, the estimator is defined and then characterized in the following.

Definition 10 ($\xi\alpha$-Estimator of the Recall) *For stable soft-margin SVMs, the $\xi\alpha$-estimator of the recall is*

$$Rec^n_{\xi\alpha}(h_\mathcal{L}) = 1 - \frac{d_{+-}}{n_+} \quad \text{with} \quad d_{+-} = |\{i : y_i = 1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \tag{5.65}$$

$$n_+ = |\{i : y_i = 1\}| \tag{5.66}$$

with ρ equals 2. $\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 4 and 3 (or 6 and 5) on the training set S_n. R^2_Δ is an upper bound on $c \leq \mathcal{K}(\vec{x}, \vec{x}') \leq c + R^2_\Delta$ for all $\vec{x}, \vec{x}'$ and some constant c.

d_{+-} is the number of positive training examples for which the inequality $(\rho\alpha_i R^2_\Delta + \xi_i) \geq 1$ holds. n_+ is the number of positive examples in the training sample. The following lemma states that d_{+-} is an upper bound on the number loo_+ of positive examples that produce a leave-one-out error.

Lemma 6 (Bound on l_{+-} for Stable Soft-Margin SVMs) *The number l_{+-} of leave-one-out errors on positive examples for stable soft-margin SVMs on the training set S_n, is bounded by*

$$l_{+-} \leq |\{i : y_i = 1 \wedge (2\,\alpha_i R^2 + \xi_i) \geq 1\}| \tag{5.67}$$

$\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 4 and 3 (or 6 and 5) on the training set S_n. R^2 is an upper bound on $\mathcal{K}(\vec{x}, \vec{x})$ and $\mathcal{K}(\vec{x}, \vec{x}') \geq 0$.

Proof *The proof of Lemma 4 is easily specialized to this case.* ▮

l_{+-} can be used to design an almost unbiased estimator of the false positive rate.

Corollary 1 (Leave-One-Out Estimator of the False Positive Rate)
The leave-one-out estimator

$$Err^n_{+loo}(h_\mathcal{L}) = \frac{l_{+-}}{n} \tag{5.68}$$

gives an almost unbiased estimate of the false positive rate $Err^{n-1}_+(h_\mathcal{L}) = \mathcal{E}(\Pr(h_\mathcal{L}(\vec{x}) = -1, y = 1 | h_\mathcal{L}))$ in the following sense.

$$\mathcal{E}(Err^n_{+loo}(h_\mathcal{L})) = \mathcal{E}(Err^{n-1}_+(h_\mathcal{L})) \tag{5.69}$$

Proof *For the loss function*

$$L_+(h(\vec{x}), y) = \begin{cases} 1 & y = 1 \wedge h(x) = -1 \\ 0 & else \end{cases} \tag{5.70}$$

the risk correspond to the false positive rates. Since Theorem 5 also holds for L_+, the result follows immediately. ▮

Using the common definition of bias to characterize the $\xi\alpha$-estimator of the recall is difficult. The recall estimate depends on the number of positive training examples in a non-linear way. The situation is even worse for the precision. Given a classification rule that classifies all examples into the negative class with probability one, the precision is not defined at all. Researchers in information retrieval have worked around this problem by differentiating between micro-averaging and macro-averaging (cf. Section 2.6.4). Macro-expectation, the analog of macro-averaging, corresponds to the conventional expected value. In micro-averaging the arguments are averaged before the function is applied. This removes the artifacts discussed above. The following definition generalizes micro-averaging to the expectation of a function.

Definition 11 (Micro-Expected Value of a Function) *The micro-expected value $\mathcal{E}_{micro}(f(X))$ of a function $f(x)$ is defined as*

$$\mathcal{E}_{micro}(f(X)) = f(\mathcal{E}(X)) \tag{5.71}$$

The random variable X can be a vector.

It is possible to define micro-expected recall and micro-expected $Rec^n_{\xi\alpha}(h_\mathcal{L})$ now. The micro-expected recall $\mathcal{E}_{micro}(Rec(H))$ is

$$\mathcal{E}_{micro}(Rec(H)) = \frac{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1|h))}{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1|h)) + \mathcal{E}(\Pr(h(\vec{x}) = -1, y = 1|h))} \tag{5.72}$$

Note that the expectation is over H, the random variable representing the hypotheses. Similarly, the micro-expected $\xi\alpha$-estimate of the recall $\mathcal{E}_{micro}(Rec^n_{\xi\alpha}(h_\mathcal{L}))$ is

$$\mathcal{E}_{micro}(Rec^n_{\xi\alpha}(h_\mathcal{L})) = 1 - \frac{\mathcal{E}(d_{+-})}{\mathcal{E}(n_+)} \tag{5.73}$$

The expectation is over training sets of size n. The bias of the $\xi\alpha$-estimator in terms of a micro-expectation is characterized by the following theorem.

Theorem 9 (Bias of the $\xi\alpha$-Estimator of the Recall) *The $\xi\alpha$-estimator of the recall $Rec^n_{\xi\alpha}(h_\mathcal{L})$ is conservatively biased in the following sense:*

$$\mathcal{E}_{micro}(Rec^n_{\xi\alpha}(h_\mathcal{L})) \leq \mathcal{E}_{micro}(Rec^{n-1}(h_\mathcal{L})) \tag{5.74}$$

Proof *Starting with the definition of micro expected recall, the following transformations lead to a more suitable form.*

$$\begin{aligned}
\mathcal{E}_{micro}(Rec(H)) &= \frac{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1|h))}{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1|h)) + \mathcal{E}(\Pr(h(\vec{x}) = -1, y = 1|h))} && (5.75)\\
&= \frac{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1|h))}{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1|h) + \Pr(h(\vec{x}) = -1, y = 1|h))} && (5.76)\\
&= \frac{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1|h))}{\mathcal{E}(\Pr(y = 1|h))} && (5.77)
\end{aligned}$$

The probability of drawing a positive example is independent of h and the training set, so $\Pr(y = 1) = \mathcal{E}(\Pr(y = 1|h))$. *Since* $\Pr(h(\vec{x}) = 1, y = 1|h) = \Pr(y = 1) - \Pr(h(\vec{x}) = -1, y = 1|h)$*, it holds that*

$$\begin{aligned}
\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1|h)) &= \mathcal{E}(\Pr(y = 1) - \Pr(h(\vec{x}) = -1, y = 1|h)) && (5.78)\\
&= \mathcal{E}(\Pr(y = 1|h)) - \mathcal{E}(\Pr(h(\vec{x}) = -1, y = 1|h)) && (5.79)\\
&= \Pr(y = 1) - \mathcal{E}(\Pr(h(\vec{x}) = -1, y = 1|h)) && (5.80)
\end{aligned}$$

It follows that the micro-expected recall (the expectation is taken over training sets of size $n-1$) is bounded from below by

$$
\begin{aligned}
\mathcal{E}_{micro}(Rec^{n-1}(h_{\mathcal{L}})) &= \frac{\mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x})=1, y=1|h_{\mathcal{L}}))}{\mathcal{E}(\Pr(y=1))} && (5.81)\\
&= \frac{\Pr(y=1) - \mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x})=-1, y=1|h_{\mathcal{L}}))}{\Pr(y=1)} && (5.82)\\
&= 1 - \frac{\mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x})=-1, y=1|h_{\mathcal{L}}))}{\Pr(y=1)} && (5.83)\\
&\geq 1 - \frac{\frac{\mathcal{E}(d_{+-})}{n}}{\Pr(y=1)} && (5.84)\\
&= 1 - \frac{\mathcal{E}(d_{+-})}{\mathcal{E}(n_+)} && (5.85)\\
&= \mathcal{E}_{micro}(Rec^n_{\xi\alpha}(h_{\mathcal{L}})) && (5.86)
\end{aligned}
$$

The inequality holds since d_{+-} is an upper bound on l_{+-} (shown in Lemma 6, possibly after invariant transformation of the kernel function according to Lemma 5), and $\mathcal{E}(l_{+-}) = \Pr(h_{\mathcal{L}}(\vec{x})=-1, y=1|h_{\mathcal{L}})$ (shown in Corollary 1). ∎

The theorem shows that the $\xi\alpha$-estimate of the recall is "on average" lower than the true recall in terms of a micro-average. Assuming that the variance of the estimate is low, $Rec^n_{\xi\alpha}(h_{\mathcal{L}})$ can be used as a lower bound on the true recall. The variance will be analyzed empirically in Section 5.4.

5.2.3 Precision

The $\xi\alpha$-estimator of the precision is defined as follows.

Definition 12 ($\xi\alpha$-Estimator of the Precision) *For stable soft-margin SVMs, the $\xi\alpha$-estimator of the precision is*

$$
\begin{aligned}
Prec^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{n_+ - d_{+-}}{n_+ - d_{+-} + d_{-+}} \quad \textit{with} \quad d_{+-} &= |\{i : y_i = 1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\}| && (5.87)\\
d_{-+} &= |\{i : y_i = -1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\} && (5.88)\\
n_+ &= |\{i : y_i = 1\}| && (5.89)
\end{aligned}
$$

with ρ equals 2. $\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 4 and 3 (or 6 and 5) on the training set S_n. R^2_Δ is an upper bound on $c \leq \mathcal{K}(\vec{x},\vec{x}') \leq c + R^2_\Delta$ for all $\vec{x},\vec{x}'$ and some constant c.

In analogy to d_{+-} the value of d_{-+} is an upper bound on the number l_{-+} of leave-one-out errors on negative training examples.

Lemma 7 (Bound on l_{-+} for Stable Soft-Margin SVMs) *The number l_{-+} of leave-one-out errors on negative examples for stable soft-margin SVM on the training sample S_n, is bounded by*

$$
l_{-+} \leq |\{i : y_i = -1 \wedge (\rho\,\alpha_i R^2 + \xi_i) \geq 1\}| \qquad (5.90)
$$

$\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 4 and 3 (or 6 and 5) on the training set S_n. R^2 is an upper bound on $\mathcal{K}(\vec{x},\vec{x})$ and $\mathcal{K}(\vec{x}_i,\vec{x}_j) \geq 0$.

Proof *The proof of Lemma 4 is easily specialized to this case.* ▮

Like the false positive rate, it is possible to get a leave-one-out estimate of the false negative rate.

Corollary 2 (Leave-One-Out Estimator for the False Negative Rate) *The leave-one-out estimator*

$$Err^n_{-loo}(h_{\mathcal{L}}) = \frac{l_{-+}}{n} \tag{5.91}$$

gives an almost unbiased estimate of the false positive rate $Err^{n-1}_{-}(h_{\mathcal{L}}) = \Pr(h_{\mathcal{L}}(\vec{x}) = 1, y = -1|h_{\mathcal{L}})$ *in the following sense.*

$$\mathcal{E}(Err^n_{-loo}(h_{\mathcal{L}})) = \mathcal{E}(Err^{n-1}_{-}(h_{\mathcal{L}})) \tag{5.92}$$

Proof *For the loss function*

$$L_{-}(h(\vec{x}), y) = \begin{cases} 1 & y = -1 \wedge h(x) = 1 \\ 0 & else \end{cases} \tag{5.93}$$

the risk correspond to the false negative rates. Since Theorem 5 also holds for L_{-}*, the result follows immediately.* ▮

Again, micro-expectation is used to characterize the bias of the estimator. The micro-expected precision $\mathcal{E}_{micro}(Prec(H))$ is

$$\mathcal{E}_{micro}(Rec(H)) = \frac{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1|h))}{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1|h)) + \mathcal{E}(\Pr(h(\vec{x}) = 1, y = -1|h))} \tag{5.94}$$

and the micro-expected $\xi\alpha$-estimate of the precision $\mathcal{E}_{micro}(Prec^n_{\xi\alpha}(h_{\mathcal{L}}))$ is

$$\mathcal{E}_{micro}(Prec^n_{\xi\alpha}(h_{\mathcal{L}})) = \frac{\mathcal{E}(n_{+}) - \mathcal{E}(d_{+-})}{\mathcal{E}(n_{+}) - \mathcal{E}(d_{+-}) + \mathcal{E}(d_{-+})} \tag{5.95}$$

Again, the $\xi\alpha$-estimator underestimates the true value "on average".

Theorem 10 (Bias of the $\xi\alpha$-Estimator of the Precision) *The $\xi\alpha$-estimator of the precision* $Prec^n_{\xi\alpha}(h_{\mathcal{L}})$ *is conservatively biased in the following sense:*

$$\mathcal{E}_{micro}(Prec^n_{\xi\alpha}(h_{\mathcal{L}})) \leq \mathcal{E}_{micro}(Prec^{n-1}(h_{\mathcal{L}})) \tag{5.96}$$

Proof *Recall from the proof of Theorem 9 that*

$$\begin{aligned} \mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x}) = 1, y = 1|h_{\mathcal{L}})) &= \Pr(y = 1) - \mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x}) = -1, y = 1|h_{\mathcal{L}})) && (5.97) \\ &\geq \Pr(y = 1) - \frac{\mathcal{E}(d_{+-})}{n} && (5.98) \end{aligned}$$

The following chain of transformations proves the theorem:

$$\begin{aligned} \mathcal{E}_{micro}(Prec^{n-1}(h_{\mathcal{L}})) &= \frac{\mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x}) = 1, y = 1|h_{\mathcal{L}}))}{\mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x}){=}1, y{=}1|h_{\mathcal{L}})) + \mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x}){=}1, y{=}{-}1|h_{\mathcal{L}}))} && (5.99) \\ &\geq \frac{\Pr(y = 1) - \frac{\mathcal{E}(d_{+-})}{n}}{\Pr(y = 1) - \frac{\mathcal{E}(d_{+-})}{n} + \mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x}) = 1, y = -1|h_{\mathcal{L}}))} && (5.100) \\ &\geq \frac{\Pr(y = 1) - \frac{\mathcal{E}(d_{+-})}{n}}{\Pr(y = 1) - \frac{\mathcal{E}(d_{+-})}{n} + \frac{\mathcal{E}(d_{-+})}{n}} && (5.101) \\ &= \frac{\mathcal{E}(n_{+}) - \mathcal{E}(d_{+-})}{\mathcal{E}(n_{+}) - \mathcal{E}(d_{+-}) + \mathcal{E}(d_{-+})} && (5.102) \\ &= \mathcal{E}_{micro}(Prec^n_{\xi\alpha}(h_{\mathcal{L}})) && (5.103) \end{aligned}$$

The first inequality holds since d_{+-} is an upper bound on l_{+-} (shown in Lemma 6, possibly after invariant transformation of the kernel function according to Lemma 5), and $\mathcal{E}(l_{+-}) = \mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x}) = -1, y = 1|h_{\mathcal{L}}))$ (shown in Corollary 1). The second inequality holds since d_{-+} is an upper bound on l_{-+} (shown in Lemma 7), and $\mathcal{E}(l_{-+}) = \mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x}) = 1, y = -1|h_{\mathcal{L}}))$ (shown in Corollary 2). ▮

5.2.4 $F1$-Measure

The following defines an estimator of the $F1$-measure.

Definition 13 ($\xi\alpha$-Estimator of the $F1$-Measure) *For stable soft-margin SVMs, the $\xi\alpha$-estimator of the $F1$-measure is*

$$F1^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{2\,n_+ - 2\,d_{+-}}{2\,n_+ - d_{+-} + d_{-+}} \quad \textit{with} \quad d_{+-} = |\{i : y_i = 1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \tag{5.104}$$

$$d_{-+} = |\{i : y_i = -1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\} \tag{5.105}$$

$$n_+ = |\{i : y_i = 1\}| \tag{5.106}$$

with ρ equals 2. $\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 4 and 3 (or 6 and 5) on the training set S_n. R^2_Δ is an upper bound on $c \leq \mathcal{K}(\vec{x}, \vec{x}') \leq c + R^2_\Delta$ for all $\vec{x}, \vec{x}'$ and some constant c.

Defining micro-expected $F1$ in analogy with precision and recall, it is again possible to show that the $\xi\alpha$-estimator is "on average" below the true value.

Theorem 11 (Bias of the $\xi\alpha$-Estimator of the $F1$-Measure) *The $\xi\alpha$-estimator of the $F1$-measure $F1^n_{\xi\alpha}(h_{\mathcal{L}})$ is conservatively biased in the following sense:*

$$\mathcal{E}_{micro}(F1^n_{\xi\alpha}(h_{\mathcal{L}})) \leq \mathcal{E}_{micro}(F1^{n-1}(h_{\mathcal{L}})) \tag{5.107}$$

Proof *The following chain of transformations proves the theorem. It starts with the micro-expectation of $F1$, and then uses corollaries 2 and 1 in combination with lemmas 6, 7, and 5.*

$$\begin{aligned}
&\mathcal{E}_{micro}(F1^{n-1}(h_{\mathcal{L}}))\\
&\quad= \frac{2\mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x})=1, y=1|h_{\mathcal{L}}))}{2\mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x})=1, y=1|h_{\mathcal{L}})) + \mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x})=-1, y=1|h_{\mathcal{L}})) + \mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x})=1, y=-1|h_{\mathcal{L}}))}\\
&\quad\geq \frac{2\,(\Pr(y=1) - \frac{\mathcal{E}(d_{+-})}{n})}{2\,(\Pr(y=1) - \frac{\mathcal{E}(d_{+-})}{n}) + \mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x})=-1, y=1|h_{\mathcal{L}})) + \mathcal{E}(\Pr(h_{\mathcal{L}}(\vec{x})=1, y=-1|h_{\mathcal{L}}))}\\
&\quad\geq \frac{2\ \Pr(y=1) - 2\,\frac{\mathcal{E}(d_{+-})}{n}}{2\ \Pr(y=1) - \frac{\mathcal{E}(d_{+-})}{n} + \frac{\mathcal{E}(d_{-+})}{n}}\\
&\quad= \frac{2\,\mathcal{E}(n_+) - 2\,\mathcal{E}(d_{+-})}{2\,\mathcal{E}(n_+) - \mathcal{E}(d_{+-}) + \mathcal{E}(d_{-+})}\\
&\quad= \mathcal{E}_{micro}(F1^n_{\xi\alpha}(h_{\mathcal{L}}))
\end{aligned}$$

▮

5.3 Fast Leave-One-Out Estimation

The efficiency of the $\xi\alpha$-estimators is bought at the expense of a conservative bias. At the other end of the spectrum, conventional leave-one-out estimators are essentially unbiased, but computationally very inefficient. Is it possible to remove the conservative bias of $\xi\alpha$-estimators at only a small increase of computational expense?

Lemma 4 direcly implies the following relationship:

$$(\rho\alpha_i R_\Delta^2 + \xi_i) < 1 \Rightarrow \left[h^{\setminus t}(\vec{x}_t) = y_t\right] \tag{5.108}$$

If $(\rho\alpha_i R_\Delta^2 + \xi_i) < 1$ holds, then the example $(\vec{x}_i, y_i)$ does not produce a leave-one-out error. It is not necessary to perform actual leave-one-out testing for such examples. The observation of $(\rho\alpha_i R_\Delta^2 + \xi_i) < 1$ after training on the complete sample is sufficient to deduce that the corresponding example will not produce a leave-one-out error.

Similarly, examples corresponding to training errors are a priori determined. The following lemma shows that each training error will also produce a leave-one-out error.

Lemma 8 *For any soft-margin SVM on a training set S_n it holds that training errors also produce leave-one-out errors.*

$$(\xi_t > 1) \Longrightarrow \left[h^{\setminus t}(\vec{x}_t) \neq y_t\right] \tag{5.109}$$

Proof *Let $\vec{w}, b, \xi_1, \ldots, \xi_n$ be the solution of the primal SVM training problem on the whole sample S_n. Similarly, let $\vec{w}^t, b^t, \xi_1^t, \ldots, \xi_{t-1}^t, \ldots, \xi_{t+1}^t, \ldots, \xi_n^t$ be the solution on the sample $S_n^{\setminus t}$ with the t-th example held out. The following chain of inequalities holds.*

$$\begin{aligned}
\frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i\neq t}\xi_i + C\xi_t &\leq \frac{1}{2}\vec{w}^t\cdot\vec{w}^t + C\sum_{i\neq t}\xi_i^t + C\left\lfloor 1 - y_t\left[\vec{w}^t\cdot\vec{x}_t + b^t\right]\right\rfloor && (5.110)\\
&\leq \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i\neq t}\xi_i + C\left\lfloor 1 - y_t\left[\vec{w}^t\cdot\vec{x}_t + b^t\right]\right\rfloor && (5.111)
\end{aligned}$$

Note that $h^{\setminus t}(\vec{x}_t)$ makes a leave-one-out error on example t exactly when $y_t\left[\vec{w}^t\cdot\vec{x}_t + b^t\right] < 0$. Using basic algebraic transformations and exploiting that $\xi_t > 1$ for training errors it follows:

$$\begin{aligned}
\frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i\neq t}\xi_i + C\xi_t &\leq \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i\neq t}\xi_i + C\left\lfloor 1 - y_t\left[\vec{w}^t\cdot\vec{x}_t + b^t\right]\right\rfloor && (5.112)\\
\Leftrightarrow \xi_t &\leq \left\lfloor 1 - y_t\left[\vec{w}^t\cdot\vec{x}_t + b^t\right]\right\rfloor && (5.113)\\
\Rightarrow 1 &< 1 - y_t\left[\vec{w}^t\cdot\vec{x}_t + b^t\right] && (5.114)\\
\Rightarrow 0 &< y_t\left[\vec{w}^t\cdot\vec{x}_t + b^t\right] && (5.115)
\end{aligned}$$

▮

The lemma shows that it is not necessary to perform actual leave-one-out testing for training errors. Both condition (5.108) and condition (5.109) can potentially identify many examples where re-training the SVM can be avoided. This puts computing the exact leave-one-out quantities l_{++}, l_{+-}, l_{-+}, and l_{--} of the leave-one-out contingency table corresponding to Definition 1 into the range of practical tractability. By how much these conditions speed up exact leave-one-out estimation is evaluated experimentally in the following section.

5.4 Experiments

The following experiments explore how well the $\xi\alpha$-estimators work in practice and by how much exact leave-one-out estimation can be sped up. The evaluation is done on the three text-classification tasks Reuters, WebKB, and Ohsumed. Unless noted otherwise, the setup described in Section 2.7 is used. The results for Reuters are discussed in detail. The findings are validated on the other two collections.

Two values for the parameter ρ are evaluated in the following, namely $\rho = 2$ and $\rho = 1$. The setting $\rho = 2$ is a direct consequence of Lemma 4, while the setting $\rho = 1$ is suggested as a better choice for text classification by the following argument. The factor $\rho = 2$ in the $\xi\alpha$-estimates was introduced to upper bound the expressions

$$\frac{1}{2}\sum_{i \in SV^{\setminus t}} \sum_{j \in SV^{\setminus t}} \mu_i \mu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \tag{5.116}$$

and

$$\frac{1}{2}\sum_{i \in SV^t} \sum_{j \in SV^t} \nu_i \nu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \tag{5.117}$$

in the proof of Lemma 4. In the worst case, each expression can be $\frac{1}{2}\alpha_t^2 R^2$ as argued above. For this worst case to happen, it is necessary that the number of support vectors is very small or that all support vectors have identical feature vectors $\vec{x}_i$. For text-classification problems the opposite is true. There are many support vectors and many of them are almost orthogonal. This means that most entries in the corresponding part of the Hesse-matrix are small or zero for the linear kernel. Consequently (5.116) and (5.117) are close to zero, leading to a $\xi\alpha$-estimate with $\rho \approx 1$ instead of $\rho = 2$. Similar arguments can be made also for non-linear kernels that are based on the dot-product. In the following, the validity of $\rho = 1$ is also verified experimentally.

Unless noted otherwise, the following results are averages over 10 random test/training splits and the variance around each average is estimated. Training and test set are designed to be of equal size, in order to be able to compare variance estimates. The $\xi\alpha$-estimators are applied to the SVM trained on the training set. The test set is used to get a hold-out estimate as an approximation to the true parameter. For simplicity reasons, all results in this section are for linear SVMs with $C = 0.5$. This value of C was selected by the $\xi\alpha$-estimates in model-selection experiments (see Chapter 6). Clearly, this is the setting of prime interest, since it produces the final classification rule. No preprocessing like stemming or stopword removal is performed and all words that occur in at least 3 training documents are used as features. These features are TFIDF weighted [Salton and Buckley, 1988]. The resulting document vectors are normalized to unit length. This implies $R_\Delta = 1$. Experiments with non-linear SVMs can be found in Chapter 6.

5.4.1 How Large are Bias and Variance of the $\xi\alpha$-Estimators?

Figure 5.1 illustrates the results for the Reuters dataset with $\rho = 1$. The findings are as expected. The $\xi\alpha$-estimators overestimate the true error and underestimate precision, recall, and $F1$ as predicted. For the 100 experiments (10 splits for 10 classes) the estimate of the error was lower than the error measured on the hold-out set in 3 cases. The estimated recall was higher in only 1, precision in 15, and $F1$ in 2 experiments. Since the average $\xi\alpha$-estimates are generally close to the average hold-out estimates, this indicates a small variance of the $\xi\alpha$-estimate, in particular, since the hold-out estimate is subject to variance, too.

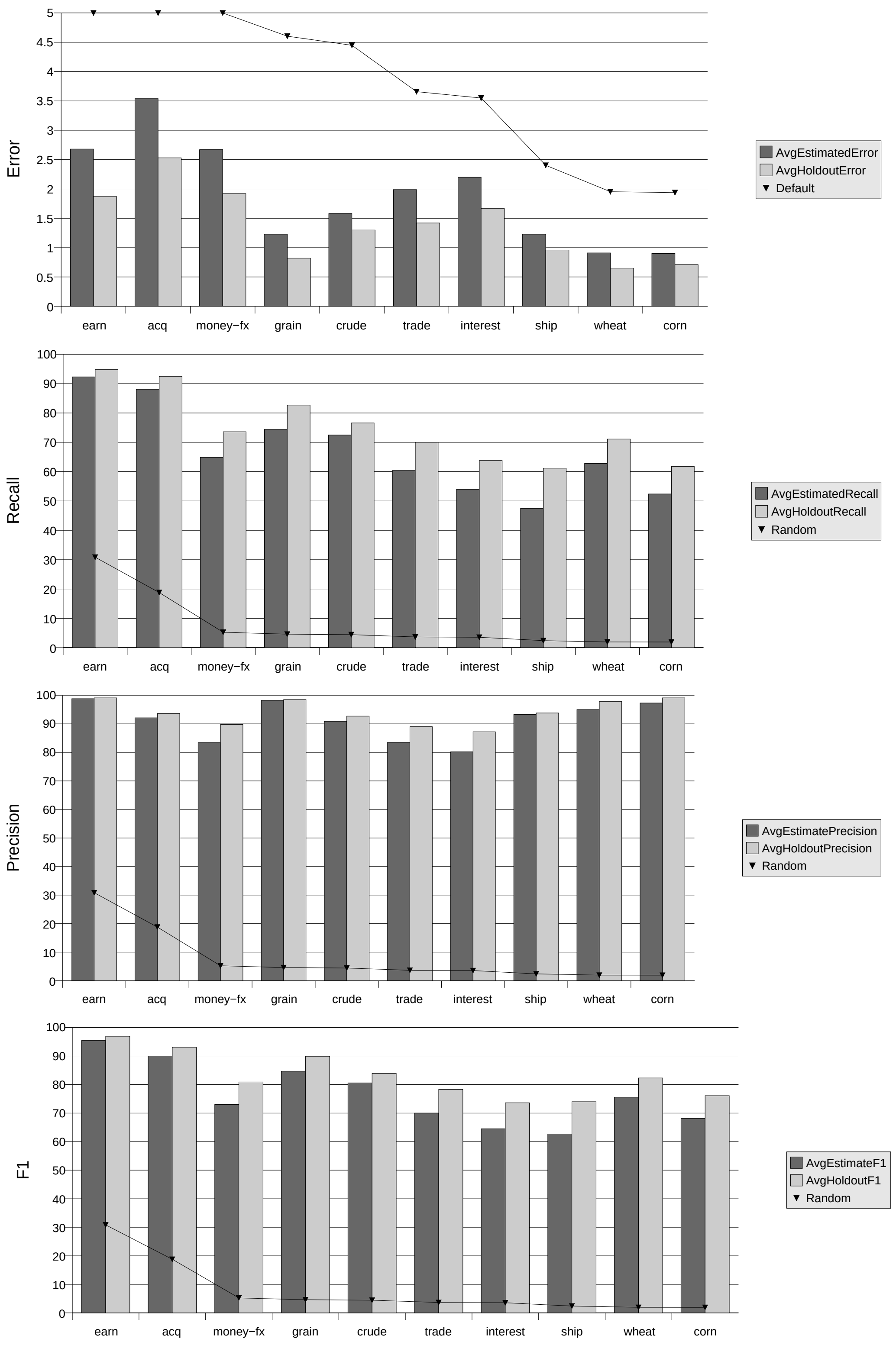


Figure 5.1: Diagrams comparing average $\xi\alpha$-estimate ($\rho = 1$) of the error, the recall, the precision, and the $F1$-measure with the average true error, true recall, true precision, and true $F1$ measured on a hold-out set for the ten most frequent Reuters categories.

			$Err_{\xi\alpha}(h)$				$F1_{\xi\alpha}(h)$	
	$Err_{test}(h)$	$Err_{loo}(h)$	$\rho = 1$	$\rho = 2$	$F1_{test}(h)$	$F1_{loo}(h)$	$\rho = 1$	$\rho = 2$
earn	1.87 ± 0.15	2.14 ± 0.08	2.68 ± 0.08	6.79 ± 0.21	96.9 ± 0.2	96.4 ± 0.1	95.4 ± 0.1	88.3 ± 0.4
acq	2.53 ± 0.20	2.54 ± 0.14	3.54 ± 0.16	12.99 ± 0.28	93.1 ± 0.4	92.9 ± 0.4	90.0 ± 0.5	62.5 ± 1.2
money-fx	1.92 ± 0.14	2.05 ± 0.10	2.67 ± 0.12	5.38 ± 0.22	80.9 ± 1.4	79.6 ± 1.3	73.0 ± 1.4	44.2 ± 1.8
grain	0.82 ± 0.11	0.95 ± 0.10	1.23 ± 0.09	3.20 ± 0.19	89.9 ± 1.2	88.5 ± 1.5	84.7 ± 1.2	57.9 ± 2.0
crude	1.30 ± 0.10	1.22 ± 0.04	1.58 ± 0.08	3.69 ± 0.20	83.9 ± 1.3	84.9 ± 0.8	80.6 ± 1.0	52.6 ± 2.1
trade	1.42 ± 0.10	1.58 ± 0.13	1.99 ± 0.11	3.80 ± 0.15	78.3 ± 1.6	76.7 ± 1.9	70.0 ± 1.7	41.4 ± 1.9
interest	1.67 ± 0.20	1.76 ± 0.17	2.20 ± 0.18	4.19 ± 0.26	73.6 ± 3.4	72.2 ± 3.7	64.5 ± 4.4	32.8 ± 4.6
ship	0.96 ± 0.13	1.06 ± 0.10	1.23 ± 0.10	2.21 ± 0.08	74.0 ± 2.9	68.8 ± 4.0	62.7 ± 5.0	26.6 ± 3.5
wheat	0.65 ± 0.08	0.71 ± 0.08	0.91 ± 0.07	1.79 ± 0.14	82.3 ± 1.4	81.7 ± 1.4	75.6 ± 1.2	52.2 ± 1.9
corn	0.71 ± 0.06	0.73 ± 0.05	0.90 ± 0.05	1.72 ± 0.12	76.1 ± 1.3	75.6 ± 1.5	68.1 ± 3.6	37.6 ± 2.2

			$Rec_{\xi\alpha}(h)$				$Prec_{\xi\alpha}(h)$	
	$Rec_{test}(h)$	$Rec_{loo}(h)$	$\rho = 1$	$\rho = 2$	$Prec_{test}(h)$	$Prec_{loo}(h)$	$\rho = 1$	$\rho = 2$
earn	94.8 ± 0.5	93.7 ± 0.3	92.3 ± 0.3	84.4 ± 0.4	99.1 ± 0.1	99.2 ± 0.1	98.8 ± 0.1	92.7 ± 0.5
acq	92.5 ± 0.5	91.3 ± 0.7	88.1 ± 0.8	59.3 ± 1.4	93.6 ± 0.7	94.6 ± 0.4	92.1 ± 0.4	66.0 ± 1.2
money-fx	73.6 ± 2.4	71.9 ± 1.8	64.9 ± 2.0	38.4 ± 2.0	89.8 ± 1.9	89.1 ± 1.6	83.4 ± 1.4	52.1 ± 1.8
grain	82.7 ± 2.1	79.9 ± 2.4	74.4 ± 1.7	48.1 ± 2.0	98.5 ± 0.5	99.1 ± 0.5	98.2 ± 0.8	72.8 ± 1.8
crude	76.6 ± 2.6	76.9 ± 1.6	72.5 ± 1.7	45.3 ± 2.1	92.7 ± 1.2	94.8 ± 0.8	90.9 ± 1.0	62.8 ± 2.4
trade	70.0 ± 2.4	67.6 ± 2.8	60.4 ± 2.1	34.7 ± 1.8	89.0 ± 1.5	88.8 ± 0.9	83.5 ± 1.3	51.2 ± 2.2
interest	63.8 ± 4.3	61.8 ± 5.1	54.0 ± 5.2	27.5 ± 4.2	87.2 ± 2.8	87.2 ± 2.3	80.2 ± 2.9	40.8 ± 5.0
ship	61.2 ± 3.9	54.0 ± 5.3	47.5 ± 5.6	18.2 ± 2.6	93.8 ± 2.5	95.6 ± 1.5	93.3 ± 1.9	49.4 ± 5.2
wheat	71.1 ± 2.3	70.4 ± 1.6	62.8 ± 1.8	43.2 ± 1.8	97.8 ± 1.0	97.5 ± 1.3	95.0 ± 1.5	65.8 ± 2.7
corn	61.8 ± 1.7	61.3 ± 2.3	52.4 ± 4.1	28.1 ± 2.1	99.1 ± 0.6	98.7 ± 0.8	97.3 ± 1.8	57.1 ± 3.2

Table 5.1: Table comparing average $\xi\alpha$-estimate and leave-one-out estimate of the error, the recall, the precision, and the $F1$ with the average true error, true recall, true precision, and true $F1$ for the ten most frequent Reuters categories. The "true" values are estimated from a hold-out set of the same size as the training set (6451 examples each). All values are averaged over 10 random test/training splits exhibiting the standard deviation printed after each average.

This is also supported by Table 5.1, which gives additional details on the results. The table contains the $\xi\alpha$-estimates with $\rho = 1$ and with $\rho = 2$, as well as the exact leave-one-out estimates. For $\rho = 2$ the $\xi\alpha$-estimates are substantially more biased than for $\rho = 1$. Yet, for $\rho = 1$ the criterion is still exact. Comparing to an exact leave-one-out via repeated retraining, over all 90 categories and 10 different test/training splits the $\rho = 1$ criterion never missed a true leave-one-out error.

The conservative bias is not entirely due to the properties of the $\xi\alpha$-estimators. Table 5.1 shows that, surprisingly, the exact leave-one-out estimates appear to be biased as well. My conjecture is that this bias is due to the experimental setup of selecting the attributes of the feature vector based on the training data. This implies that training and test vectors are not strictly i.i.d.

After each average, the table includes an estimate of the standard deviation. The standard deviation of the $\xi\alpha$-estimates is very similar to that of the leave-one-out estimates, especially for $\rho = 1$. This shows that, in terms of variance, the $\xi\alpha$-estimates are as good as exact leave-one-out.

To make sure that the $\xi\alpha$-estimators are not tailored to the properties of the Reuters dataset, but apply to a wide range of text-classification tasks, similar experiments were conducted also for the WebKB dataset (Table 5.2) and the Ohsumed data (Table 5.3). For both collections, the results are qualitatively the same as for Reuters.

The conclusion from this experiment is that for text classification the $\xi\alpha$-estimates with $\rho = 1$ are preferable over those with $\rho = 2$. The $\xi\alpha$-estimates with $\rho = 1$ exhibit a moderate conservative bias and a variance equivalent to the exact leave-one-out estimate.

			$Err_{\xi\alpha}(h)$				$F1_{\xi\alpha}(h)$	
	$Err_{test}(h)$	$Err_{loo}(h)$	$\rho = 1$	$\rho = 2$	$F1_{test}(h)$	$F1_{loo}(h)$	$\rho = 1$	$\rho = 2$
course	2.43 ± 0.34	2.75 ± 0.26	3.48 ± 0.30	11.93 ± 0.47	94.2 ± 0.8	93.4 ± 0.7	91.6 ± 0.7	68.9 ± 1.4
faculty	9.80 ± 0.33	11.19 ± 0.51	13.40 ± 0.64	31.98 ± 0.78	79.0 ± 0.7	74.7 ± 1.4	68.9 ± 1.8	28.4 ± 1.9
project	7.75 ± 0.34	8.69 ± 0.40	9.56 ± 0.46	14.80 ± 0.23	53.3 ± 3.1	44.4 ± 3.8	35.9 ± 4.8	3.6 ± 1.4
student	7.64 ± 0.36	7.95 ± 0.42	10.14 ± 0.59	32.42 ± 0.50	89.9 ± 0.4	89.5 ± 0.6	86.4 ± 0.8	55.9 ± 0.8

			$Rec_{\xi\alpha}(h)$				$Prec_{\xi\alpha}(h)$	
	$Rec_{test}(h)$	$Rec_{loo}(h)$	$\rho = 1$	$\rho = 2$	$Prec_{test}(h)$	$Prec_{loo}(h)$	$\rho = 1$	$\rho = 2$
course	90.7 ± 1.2	88.7 ± 1.0	85.9 ± 1.2	60.0 ± 1.8	98.0 ± 0.5	98.7 ± 0.4	98.0 ± 0.3	81.0 ± 1.2
faculty	69.2 ± 1.0	61.8 ± 1.7	55.7 ± 2.2	23.8 ± 1.7	92.2 ± 1.3	94.5 ± 1.3	90.5 ± 1.2	35.3 ± 2.2
project	37.0 ± 3.1	29 ± 3.1	22.4 ± 3.6	2.3 ± 0.9	95.9 ± 1.2	95.2 ± 1.5	91.3 ± 2.5	8.2 ± 3.0
student	87.0 ± 0.4	86.1 ± 1.0	82.3 ± 0.9	52.5 ± 1.1	93.0 ± 0.6	93.2 ± 0.8	90.9 ± 0.7	59.8 ± 0.6

Table 5.2: Same as Table 5.1, but for the WebKB dataset. The training/test sets contain 2092 examples.

			$Err_{\xi\alpha}(h)$				$F1_{\xi\alpha}(h)$	
	$Err_{test}(h)$	$Err_{loo}(h)$	$\rho = 1$	$\rho = 2$	$F1_{test}(h)$	$F1_{loo}(h)$	$\rho = 1$	$\rho = 2$
Pathology	18.56 ± 0.36	18.79 ± 0.24	20.96 ± 0.22	33.33 ± 0.50	39.1 ± 1.4	37.5 ± 1.7	29.5 ± 1.6	9.3 ± 0.7
Cardiovascular	7.13 ± 0.15	7.23 ± 0.21	8.73 ± 0.24	15.41 ± 0.25	74.1 ± 0.7	73.0 ± 1.3	66.7 ± 1.3	41.7 ± 1.1
Neoplasms	5.42 ± 0.14	5.56 ± 0.13	6.51 ± 0.15	11.77 ± 0.21	78.8 ± 0.6	78.2 ± 0.7	73.8 ± 0.7	52.1 ± 0.9
Nervous System	7.31 ± 0.15	7.42 ± 0.22	8.41 ± 0.21	13.22 ± 0.19	53.6 ± 1.2	51.2 ± 2.3	42.8 ± 2.5	15.5 ± 1.3
Immunologic	5.28 ± 0.22	5.33 ± 0.14	6.04 ± 0.19	9.37 ± 0.29	60.5 ± 1.0	59.4 ± 1.0	52.3 ± 1.1	27.3 ± 1.3

			$Rec_{\xi\alpha}(h)$				$Prec_{\xi\alpha}(h)$	
	$Rec_{test}(h)$	$Rec_{loo}(h)$	$\rho = 1$	$\rho = 2$	$Prec_{test}(h)$	$Prec_{loo}(h)$	$\rho = 1$	$\rho = 2$
Pathology	25.8 ± 1.3	24.3 ± 1.5	18.9 ± 1.2	7.4 ± 0.6	81.2 ± 2.1	82.0 ± 0.9	67.0 ± 1.0	12.6 ± 0.9
Cardiovascular	64.0 ± 0.8	62.0 ± 1.7	55.3 ± 1.5	34.9 ± 1.1	88.0 ± 0.8	88.9 ± 0.6	84 ± 1.0	51.8 ± 1.1
Neoplasms	67.3 ± 0.8	66.0 ± 1.0	60.7 ± 1.0	42.4 ± 1.0	95.2 ± 0.6	95.8 ± 0.4	94.0 ± 0.4	67.6 ± 1.0
Nervous System	38.3 ± 1.3	36.0 ± 2.4	29.0 ± 2.3	11.2 ± 1.0	89.6 ± 1.5	89.4 ± 0.8	82.0 ± 1.1	25.4 ± 1.8
Immunologic	45.4 ± 1.2	44.0 ± 1.0	37.5 ± 1.1	19.8 ± 1.1	90.8 ± 1.0	91.3 ± 0.8	86.8 ± 1.2	43.6 ± 1.6

Table 5.3: Same as Table 5.1, but for the Ohsumed dataset and a training/test set size of 10000.

5.4.2 What is the Influence of the Training Set Size?

All results presented so far were for large training sets containing more than 2000 examples. Do the estimators work for smaller training sets as well? Figures 5.2 to 5.4 show learning curves for the Reuters categories "earn", "acq", and "money-fx". To save space, only error rate and $F1$ are plotted, since precision and recall behave similarly to $F1$. The top two curves of each graph show the average $\xi\alpha$-estimate and the average hold-out estimate on an additional test set of the same size as the training set. The averages are over 20 random training/test splits. Except for very small training sets, the graphs show no strong systematic connection between bias (i. e. the difference between the top two curves of each graph) and the training set size. For small training sets, the SVM behaves almost like the default classifier for "acq" and "money-fx". In this situation the average $\xi\alpha$-estimate and the hold-out estimate are almost equal. In terms of variance, the training set size has a strong influence on both the hold-out estimate and the $\xi\alpha$-estimate. The bottom two curves of each graph show the empirical standard deviation of each estimator. As expected, the variance increases with decreasing training set size. Nevertheless, when moving to very small training sets, the variance decreases again. This is a consequence of the SVM behaving more and more like the default classifier. Interestingly, the variance curves of the $\xi\alpha$-estimator are very similar to those of the hold-out estimator.

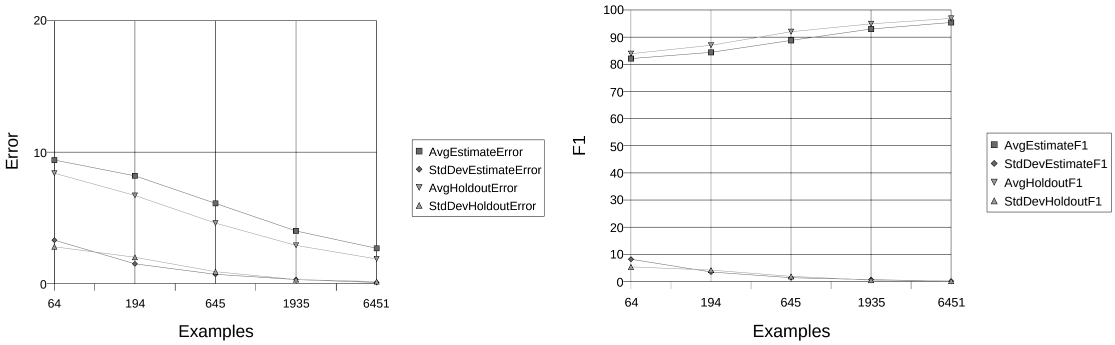


Figure 5.2: Learning curves for the Reuters category "earn" comparing the $\xi\alpha$-estimator of the error rate (left) and the $F1$ (right) with hold-out testing. The x-axis denotes the size of the training set on a log-scale. Each test set for hold-out testing contains as many examples as the corresponding training set. All values are averages over ten random test/training splits. The upper curves show the average, the lower curves show the standard deviation. Individual data points are connected to improve readability.

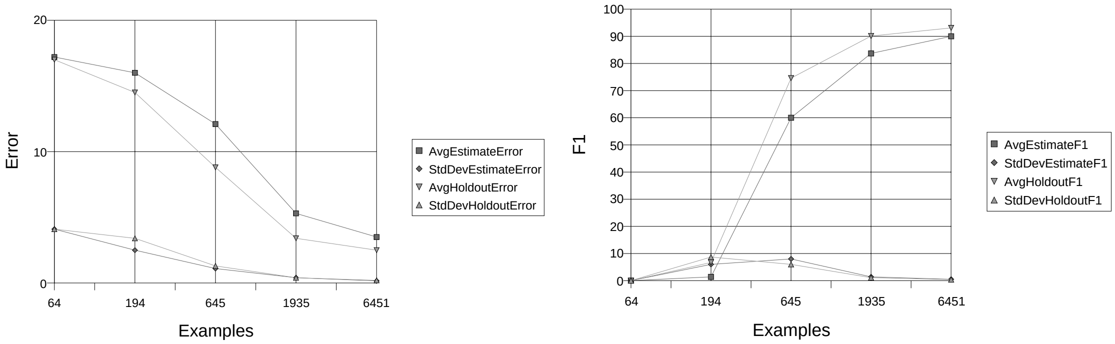


Figure 5.3: Same as Figure 5.2, but for the Reuters category "acq".

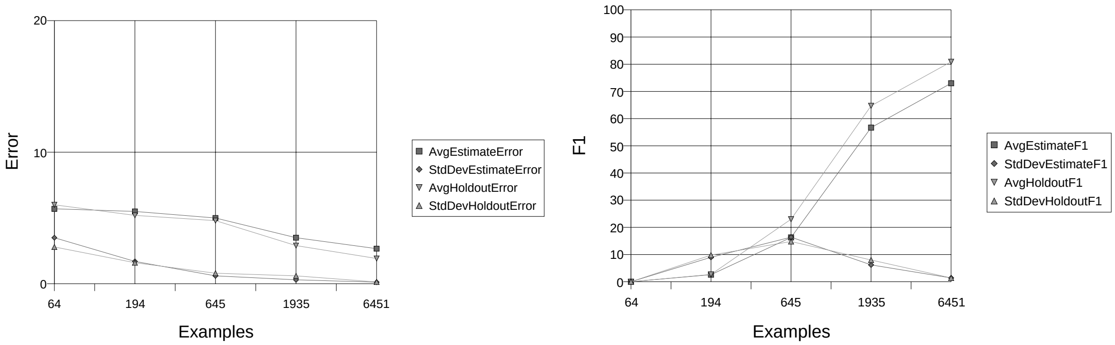


Figure 5.4: Same as Figure 5.2, but for the Reuters category "money-fx".

	cpu-time (sec)		retraining steps (%)	
	$\rho = 1$	$\rho = 2$	$\rho = 1$	$\rho = 2$
Reuters	11.1	32.3	0.20%	0.58%
WebKB	78.5	235.4	6.78%	20.42%
Ohsumed	433.0	1132.3	1.07%	2.56%

Table 5.4: CPU-time and percentage of leave-one-out retraining steps that actually need to be performed averaged over all categories. The training set size is 6451 for Reuter, 2092 for WebKB, and 10000 for Ohsumed. The value of C is 0.5 for a linear SVM.

5.4.3 How Large is the Efficiency Improvement for Exact Leave-One-Out?

Section 5.3 proposed how exact leave-one-out estimation can be sped up for support vector machines. The results in Table 5.4 show by how much efficiency can be improved compared to the standard brute-force method. The performance gain is largest for $\rho = 1$. Averaged over all the Reuters categories, only 0.2% of the retraining steps need to be computed (i.e. for 99.8% of the training examples the criteria from Section 5.3 imply the result without retraining). For 6451 Reuters articles in the training set, the exact leave-one-out quantities can now be computed in 11.1 CPU-seconds on average using a Sun Ultra/400Mhz. The performance gains on the other collections are smaller, but still substantial.

With $\rho = 2$ the criterion is more conservative about ruling out leave-one-out errors. This is reflected in a smaller speed-up. However, the criterion with $\rho = 1$ is sufficient for text. There was not a single case among all collections and all test/training splits, where the criterion with $\rho = 1$ failed to identify a leave-one-out error. This verifies that $\rho = 1$ is appropriate for text classification.

5.5 Summary and Conclusions

This chapter explores two approaches to estimating the generalization performance of an SVM with varying degree of computational efficiency.

The first approach leads to estimators that do not require any computation intensive resampling. These new $\xi\alpha$-estimators are much more efficient than cross-validation or bootstrap, since they can be computed immediately from the form of the hypothesis returned by the SVM. Moreover, the $\xi\alpha$-estimators developed here address the special measures used to evaluate text-classification performance. They can not only be used to estimate the error rate, but also the recall, the precision, and the $F1$. A theoretical analysis of the estimators shows that they tend to be conservative. This is a desirable property for practical applications, since they are less likely to falsely predict a high generalization performance. In addition to the theoretical analysis, the bias and the variance of the estimates are evaluated experimentally on three text-classification collections. As predicted by the theory, the empirical results show a conservative bias for all $\xi\alpha$-estimators. Typically, the bias is acceptably low and the variance of the $\xi\alpha$-estimates is essentially as low as that of a leave-one-out estimator.

While a conservative bias is tolerable for many tasks, this chapter also explores a second approach that removes the bias at only a moderate computational expense. The criteria that lead to the $\xi\alpha$-estimators can also be used to speed up computing the exact leave-one-out quantities of the SVM. This makes it possible to perform exact leave-one-out estimation even on large data sets. Depending on the application, this gives the option of removing the bias when computational efficiency is less important.

In Chapter 6 the $\xi\alpha$-estimators and the exact leave-one-out estimators are successfully used to perform automatic model and parameter selection. In other work it was shown that $\xi\alpha$-estimators can be used to efficiently and effectively detect concept drift [Klinkenberg and Joachims, 2000].

Part II

Methods

Chapter 6

Inductive Text Classification

After giving the theoretical motivation and justification for the maximum-margin approach to text classification in the previous two chapters, this chapter evaluates its empirical performance. It also addresses practical issues related to selecting a good representation and an appropriate parameter setting.

In particular, the following explores the use of inductive SVMs for text classification. The alternative scenario of transduction is treated in Chapter 7. The goal in inductive learning is to infer a general classification rule from a sample of labeled training documents. This classification rule should classify new examples with high accuracy. Inductive text classification is a two step process. In the first step the learner uses the training data to induce a classification rule. This step is commonly called the learning phase. In the classification phase, the second step, the classification rule is repeatedly used to classify new examples. Typical inductive text-classification tasks are the following:

EMail Routing: At a service hotline there can be multiple experts specialized in different fields. Each incoming email message should be routed to the appropriate expert by content. This is a multi-class text-classification task. As soon as a new message arrives, it has to be classified. The training data consists of previous messages labeled by who gave a competent answer.

Call Center Support: Based on speech recognition or by transcript of the user's problem, text classification can identify answers to commonly asked questions [Hammond et al., 1995][Busemann et al., 2000]. Again, the classification has to be done in real-time and for each individual query. The classification rule can be learned from previous question/answer pairs.

Browsing Assistants: Web agents like Letizia [Lieberman, 1995] and WebWatcher [Joachims et al., 1997][Mladenić, 1998][Joachims and Mladenič, 1998] assist users browsing the web. When the user comes to a new page, the assistant may e.g. highlight particularly interesting hyperlinks. This can again be cast as a text-classification problem based on the anchor texts. As soon as the user enters a new page, the classification rule must provide a decision. This rule can be learned by observing which hyperlinks the user followed on previous pages.

In this chapter I will show that SVMs are an effective and robust method for learning text classifiers from examples. The advantage of SVMs is twofold. First, they show substantially improved generalization performance over conventional text-classification techniques. Second, the SVM is robust over a wide range of parameters and does not require a statistical feature selection step. Furthermore, to select between parameters and design choices such as word-weighting scheme, use of stemming, and model parameters automatically, I will propose and evaluate model-selection methods that exploit special properties of SVMs for increased efficiency. This is particularly important if the SVM has to works autonomously without manual expert interventions and with limited computational resources.

This chapter is structured as follows. First, the formal definition of inductive text classification is reviewed. Based on the theoretical results of Chapter 5, Section 6.2 then proposes two methods for automatic model and parameter selection. They are evaluated experimentally on three test collections in Section 6.3. The performance of the resulting models is then compared to the results of conventional learning methods for text classification. The automatically selected models substantially outperforms conventional methods, giving state-of-the-art performance.

6.1 Learning Task

The goal in inductive text classification is to infer a general classification rule from a sample of labeled training documents. This classification rule should classify new examples with maximum accuracy. More formally, the learner $\mathcal{L}$ is given a training set S of n examples

$$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n) \tag{6.1}$$

drawn i.i.d. from an unknown distribution $\Pr(\vec{x}, y)$ describing the classification task. Each example consists of a feature vector $\vec{x}$ and a class label y. Here, the standard bag-of-words representation (see Section 2.2) is used for $\vec{x}$. For the label y, let us assume for simplicity that it takes only the values $+1$ and -1. Non-binary classification tasks can be handled as a series of binary classifications (see Section 2.1.2). Using the training sample S, the learner $\mathcal{L}$ aims to find a classification rule $h_{\mathcal{L}} = \mathcal{L}(S)$ that maximizes a given performance measure. This measure typically depends on $\Pr(\vec{x}, y)$ and cannot be computed directly. Commonly, (one minus) the probability of error on new examples is used as a performance measure.

$$Err(h_{\mathcal{L}}) = \Pr(h_{\mathcal{L}}(\vec{x}) \neq y | h_{\mathcal{L}}) = \int L_{0/1}(h_{\mathcal{L}}(\vec{x}), y) d\Pr(\vec{x}, y) \tag{6.2}$$

However, in text classification, precision, recall, or $F1$ on unseen test data are often more suitable and replace or augment the evaluation by error rate alone. In the following, the precision/recall breakeven point (PRBEP) (see Section 2.6) is used as the evaluation criterion.

6.2 Automatic Model and Parameter Selection

Designing the learning task is a crucial step in applying machine learning successfully. It is necessary to e.g. select an appropriate representation, a matching hypothesis space, and a good parameter setting of the learner. These design choices model essential prior knowledge about the learning task.

For text classification, there is already a fairly small set of design choices that were proven useful across different tasks. However, while text-classification problems are a somewhat restricted class of learning tasks, they still offer some variability. It is not clear if there is a single representation and parameterization that will work well in every situation. Representational changes, like the use of term weighting, can make a substantial difference in generalization performance and learning speed. Furthermore, depending on the level of noise in the data, it is necessary to set (pruning/regularization) parameters appropriately. To successfully apply machine learning to text classification in real-world applications, we therefore need methods that select an appropriate model and good parameter settings for the particular task at hand. The term *model selection* (see e.g. [Forster, 2000]) refers to the problem of selecting good learning parameters from a small set of choices based on training data. Such model-selection methods have to work under the following set of constraints.

AUTONOMY: The model-selection method needs to be formalized to a level that allows applying it without manual interventions. Especially with learning capabilities built into desktop applications like email readers or word processors, parameter tuning through a human expert is intractable.

DATA EFFICIENCY: Training data is expensive and the model-selection algorithm should make efficient use of it.

COMPUTATIONAL EFFICIENCY: The model-selection algorithm must be able to work under restricted computational resources and time constraints.

Model-selection methods are usually based on estimates or bounds on the generalization performance. This chapter compares two model-selection strategies. The first one is based on the leave-one-out method. Leave-one-out is known to provide good estimates at high computational expense. The second model-selection strategy uses the $\xi\alpha$-estimators introduced in Chapter 5. These estimators can be computed at essentially no extra cost besides training a single SVM. However, they are known to incur a conservative bias. The following defines and evaluates the corresponding model-selection strategies.

6.2.1 Leave-One-Out Estimator of the PRBEP

The leave-one-out procedure [Stone, 1974][Lunts and Brailovskiy, 1967] can be used to get an estimate

	label $y = +1$	label $y = -1$
pediction $h(\vec{x}) = +1$	l_{++}	l_{+-}
pediction $h(\vec{x}) = -1$	l_{-+}	l_{--}

of the contingency table (see Section 2.6). Leave-one-out estimation proceeds as follows. From the training sample $S = ((\vec{x}_1, y_1), \cdots, (\vec{x}_n, y_n))$ the first example $(\vec{x}_1, y_1)$ is removed. The resulting sample $S^{\backslash 1} = ((\vec{x}_2, y_2), \cdots, (\vec{x}_n, y_n))$ is used for training, leading to a classification rule $h_{\mathcal{L}}^{\backslash 1}$. This classification rule is tested on the held-out example $(\vec{x}_1, y_1)$. If the example is classified incorrectly it is said to produce a leave-one-out error. This process is repeated for all training examples. The frequencies of correct and false predictions are recorded in the contingency table.

Based on this contingency table, the leave-one-out estimates of the precision and recall are

$$Rec_{loo}^{n}(h_{\mathcal{L}}) = \frac{l_{++}}{l_{++} + l_{-+}} \tag{6.3}$$

$$Prec_{loo}^{n}(h_{\mathcal{L}}) = \frac{l_{++}}{l_{++} + l_{+-}} \tag{6.4}$$

It is also possible to design estimators for combined measures like $F1$ and the $PRBEP$. While this is straightforward for $F1$ (see Chapter 5), the $PRBEP$ requires interpolation.

Following the common approach of approximating the $PRBEP$ by the arithmetic mean of precision and recall (i.e. $PRAVG$) [Dumais et al., 1998], one comes to the following estimator

$$PRAVG^n_{loo}(h_{\mathcal{L}}) = \frac{1}{2}\left(Rec^n_{loo}(h_{\mathcal{L}}) + Prec^n_{loo}(h_{\mathcal{L}})\right) \tag{6.5}$$

This estimator is used in the following experiments. The estimates are computed using the algorithm described in Section 5.3.

6.2.2 $\xi\alpha$-Estimator of the PRBEP

While the leave-one-out estimates are usually very accurate, they are expensive to compute. For a general learner and a training sample of size n, one must run the learner n times. While the algorithm described in Section 5.3 exploits special properties of SVMs to greatly speed up this process, computing the exact leave-one-out can still be too inefficient in many practical applications. $\xi\alpha$-estimators overcome this problem by using an upper bound on the number of leave-one-out errors instead of calculating them brute force. Unlike the leave-one-out, $\xi\alpha$-estimates can be computed after training a single SVM at essentially no extra cost. However, the improved computational efficiency is traded against a bias.

$\xi\alpha$-estimators owe their name to the two arguments they are computed from. $\vec{\xi}$ is the vector of training losses at the solution of the primal SVM training problem. $\vec{\alpha}$ is the solution of the dual SVM training problem. Based on these two vectors — both are available after training the SVM at no extra cost — the $\xi\alpha$-estimators are defined using the following two counts. With R^2_Δ being the maximum difference of any two elements of the Hessian, (i.e. $c \geq \mathcal{K}(\vec{x}, \vec{x}') \leq c + R^2_\Delta$),

$$d_{+-} = |\{i : y_i = 1 \land (\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \tag{6.6}$$

counts the number of positive training examples, for which the quantity $\alpha_i R^2_\Delta + \xi_i$ exceeds one. Similarly,

$$d_{-+} = |\{i : y_i = -1 \land (\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \tag{6.7}$$

counts the respective number of negative training examples. It is proven in Chapter 5 that d_{+-} is an approximate upper bound on the number l_{-+} of leave-one-out errors on positive examples. Analogously, d_{-+} is an approximate upper bound on the number l_{+-} of negative leave-one-out errors.

It is now possible to define $\xi\alpha$-estimators for precision and recall. With n as the total number of training examples and n_+ as the number of positive training examples, the estimators are

$$Rec^n_{\xi\alpha}(h_{\mathcal{L}}) = 1 - \frac{d_{+-}}{n_+} \tag{6.8}$$

$$Prec^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{n_+ - d_{+-}}{n_+ - d_{+-} + d_{-+}} \tag{6.9}$$

Like for the leave-one-out, the $PRBEP$ is approximated by the arithmetic mean of precision and recall. The following summarizes the $\xi\alpha$-estimator of the $PRAVG$.

$$PRAVG^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{1}{2}\left(1 - \frac{d_{+-}}{n_+} + \frac{n_+ - d_{+-}}{n_+ - d_{+-} + d_{-+}}\right) \tag{6.10}$$

The theoretical properties of the $\xi\alpha$-estimators are discussed in Chapter 5.

6.2.3 Model-Selection Algorithm

The resulting model-selection algorithm — used here for maximizing $PRAVG$ as an approximation to $PRBEP$ — is straightforward and does not differ from those using other estimation methods. Input to the algorithm is a finite set $\mathcal{M}$ of models (parameter values, different representations) and a training sample S_n.

```
• for each model $m_i \in \mathcal{M}$
    – train SVM on $S_n$ using model $m_i$
    – calculate $PRAVG^n_{\xi\alpha}(h_\mathcal{L})$ / $PRAVG^n_{loo}(h_\mathcal{L})$ and store as $p_i$
• output model $m_i$ for which $p_i$ is highest.
```

The algorithm ouputs the model with the highest performance estimate p_i. Note that even if p_i is unbiased for m_i, $\max\{p_i\}$ is not an unbiased estimate of the performance of the model output by the selection algorithm.

6.3 Experiments

The experiments show the complete learning and model-selection process for three different test collections — namely Reuters, WebKB, and Ohsumed, using the setup described in Section 2.7. To improve computational efficiency, a two step approach is chosen.

In a first model-selection step, an appropriate representation is chosen. Design choices are all combinations of word-weighting method (binary, raw term frequency, and TFIDF weighting from Section 2.4), use of stemming, and use of stopword removal. In the second step the representation remains fixed, while other parameters of the SVM are selected.

The following experiments are performed for systematic test/training splits. On the Reuters task, this is the ModApte setup. It splits documents into those before and after a certain date. On the WebKB data the split is done by source. All pages from Cornell University are used for testing, while the other pages are used for training. Please note that this realistic setup makes model selection more difficult than with random subsampling. The classification task is likely to change over time and pages from different universities underlie different corporate design formats. Nevertheless, this setup is preferable to random subsampling since it better reflects the true challenges in real-world applications.

6.3.1 Word Weighting, Stemming and Stopword Removal

Table 6.1 compares $\xi\alpha$-estimates and the leave-one-out for selecting the representation. The choice is between all 12 combinations of of word-weighting method, stopword removal, and stemming using a linear SVM with $C = 1000$. An individual representation is selected for each category. The resulting test set performance is macro-averaged over all categories.

The column "random" provides a baseline. It shows the macro-averaged performance over all categories and models for each collection. The ten most frequent categories for the Reuters data sets are also evaluated separately. This is done to get some insight into the relative behavior of the model-selection methods on high and low-frequency categories. The columns $PRAVG_{\xi\alpha}$ and $PRAVG_{loo}$ show the macro-averaged performance of the model selected by the respective model-selection criterion. On all collections, the model selected by both leave-one-out and the $\xi\alpha$-estimates achieve a higher macro-averaged predictive performance on the test set than the random baseline. Using a paired t-test on the mean shows that the difference is significant for Reuters and Ohsumed. Avoiding

selecting word weighting,	macroavg. $PRAVG_{test}$ of selection by			
stemming, and stopword removal	random	$PRAVG_{\xi\alpha}$	$PRAVG_{loo}$	optimum
Reuters (top 10)	81.2	82.0	82.9	84.6
Reuters (all 90)	50.5	54.7	55.2	62.9
WebKB	81.3	84.6	85.3	87.7
Ohsumed	61.2	65.3	65.8	66.5

Table 6.1: Relative performance of model-selection methods for selecting the word-weighting scheme, the choice of stemming, and the use of stopword removal. All performance scores are macro-averages of the precision/recall average. The table compares random selection to the $\xi\alpha$-estimate and the leave-one-out estimate. The column optimum shows the average over the highest $PRAVG_{test}$ on the test set.

stemming		no			yes			no			yes		
stoplist		no			no			yes			yes		
weighting		bin	tf	tfidf	bin	tf	tfidf	bin	tf	tfidf	bin	tf	tfidf
Reuters (top 10)	macroavg. $PRAVG_{test}$	80.5	80.6	83.3	80.8	80.0	82.5	80.8	80.7	82.7	80.9	80.2	81.9
	selection by $PRAVG_{\xi\alpha}$	0	0	3	0	0	1	0	0	4	0	0	2
	selection by $PRAVG_{loo}$	0	0	3	0	0	2	0	0	3	0	1	1
Reuters (all 90)	macroavg. $PRAVG_{test}$	44.3	48.3	55.6	44.3	49.6	56.2	45.4	51.1	55.9	45.9	52.3	57.4
	selection by $PRAVG_{\xi\alpha}$	3	2	19	2	2	12	5	2	18	3	8	14
	selection by $PRAVG_{loo}$	3	4	11	3	5	15	3	9	10	4	10	13
WebKB	macroavg. $PRAVG_{test}$	85.3	86.4	78.3	84.9	82.2	76.6	81.4	84.6	78.8	79.5	81.9	76.8
	selection by $PRAVG_{\xi\alpha}$	2	0	0	0	0	0	2	0	0	0	0	0
	selection by $PRAVG_{loo}$	4	0	0	0	0	0	0	0	0	0	0	0
Ohsumed	macroavg. $PRAVG_{test}$	56.5	60.8	65.9	55.9	60.7	65.6	57.2	61.4	66.1	56.9	60.9	65.8
	selection by $PRAVG_{\xi\alpha}$	0	0	5	0	0	1	0	0	14	0	0	3
	selection by $PRAVG_{loo}$	0	1	8	0	0	0	0	0	12	0	0	2

Table 6.2: Selections of the model-selection criteria for the word-weighting scheme, the choice of stemming, and the use of stopword removal. The counts indicate for how many categories each model was selected. The macro-average of $PRAVG_{test}$ illustrates the average model performance.

the parametric assumption, binomial tests[1] also show that the estimators can select a representation significantly better than random. Note that the WebKB task involves only four categories, resulting in a small sample for significance testing.

The column "optimum" corresponds to an "unfair" model-selection criterion that uses information from the test set. It shows the macro-averaged performance for the model that gives the best results on the test set. This number is reported to give an upper bound on the performance of any model-selection algorithm. The largest difference between leave-one-out selection and the optimum occurs for the Reuters collection. Since the ten most frequent categories do not exhibit this behavior, it appears that low-frequency categories are particularly difficult. This behavior is plausible, since the low number of positive examples leads to an increased variance of the $PRAVG$ estimates.

Some insight into which representations are selected by the model-selection algorithms gives Table 6.2. For each choice of representation the table shows the macro-averaged per-

[1] For each category, the test ranks the models by their performance on the test set. Then it compares the rank of the selected model against random selection.

selecting C	macroavg. $PRAVG_{test}$ of selection by			
	random	$PRAVG_{\xi\alpha}$	$PRAVG_{loo}$	optimum
Reuters (top 10)	75.3	83.7	85.0	85.8
Reuters (all 90)	38.6	52.4	55.5	58.2
WebKB	79.7	87.6	86.3	88.9
Ohsumed	52.3	60.4	64.9	67.1

Table 6.3: Relative performance of model-selection methods for selecting the value of C. All performance scores are macro-averages of the precision/recall average. The table compares random selection to the $\xi\alpha$-estimate and the leave-one-out estimate. The column optimum shows the average over the highest $PRAVG_{test}$ on the test set.

C		0.05	0.1	0.5	1.0	5	10	1000
Reuters (top 10)	macroavg. $PRAVG_{test}$	41.9	64.2	83.7	85.1	84.5	84.6	83.3
	selection by $PRAVG_{\xi\alpha}$	0	0	10	0	0	0	0
	selection by $PRAVG_{loo}$	0	0	0	6	3	0	1
Reuters (all 90)	macroavg. $PRAVG_{test}$	4.7	10.5	38.8	46.2	57.4	57.3	55.6
	selection by $PRAVG_{\xi\alpha}$	3	3	40	11	10	13	10
	selection by $PRAVG_{loo}$	3	3	4	22	20	19	19
WebKB	macroavg. $PRAVG_{test}$	59.9	63.6	87.6	88.8	86.2	86.4	85.3
	selection by $PRAVG_{\xi\alpha}$	0	0	4	0	0	0	0
	selection by $PRAVG_{loo}$	0	0	0	2	2	0	0
Ohsumed	macroavg. $PRAVG_{test}$	9.2	34.7	59.6	64.0	66.2	66.1	66.1
	selection by $PRAVG_{\xi\alpha}$	0	2	18	3	0	0	0
	selection by $PRAVG_{loo}$	0	0	0	12	7	2	2

Table 6.4: Selections of the model-selection criteria for the value of C. The counts indicate for how many categories each model was selected. The macro-average of $PRAVG_{test}$ illustrates the average model performance.

formance and for how many categories it is selected. If the maximum of the performance estimates is not a unique, the frequency count is equally distributed over the respective selections and their test set performance is averaged. For Reuters and Ohsumed, both model-selection methods clearly prefer TFIDF weighting. For WebKB, they agree that a binary representation is preferable. The selections appear reasonable considering their average performance given in Table 6.2.

Regarding stemming and stopword removal, both methods most frequently select stopword removal, but no stemming for the Ohsumed corpus. The preference is less clear for the other two collections. For WebKB, the $\xi\alpha$-estimators tie between using a stop list or not. On Reuters, stemming and stopword removal is selected with fairly equal frequency by both estimators.

To keep the following evaluation simple, the representation most frequently selected by the $\xi\alpha$-estimators will be used for all categories of the collection. This means that TFIDF-weighting without stemming and stopword removal is used for Reuters, a binary representation with no stemming and no stopword removal is used for WebKB, and TFIDF-weighting with a stop-list is used for the Ohsumed categories.

selecting RBF-kernel γ	macroavg. $PRAVG_{test}$ of selection by			
	random	$PRAVG_{\xi\alpha}$	$PRAVG_{loo}$	optimum
Reuters (top 10)	79.2	84.1	84.1	84.7
Reuters (all 90)	48.5	54.4	53.5	57.3
WebKB	85.2	86.5	87.6	88.6
Ohsumed	59.1	65.9	66.3	67.2

Table 6.5: Relative performance of model-selection methods for selecting the parameter γ of the RBF-kernel. All performance scores are macro-averages of the precision/recall average. The table compares random selection to the $\xi\alpha$-estimate and the leave-one-out estimate. The column optimum shows the average over the highest $PRAVG_{test}$ on the test set.

6.3.2 Trading Off Training Error vs. Complexity

The parameter C in Optimization Problem 3 adjusts the amount of training error tolerated by the SVM. Using the representations selected in the previous step, Table 6.3 shows the results for selecting a good value of C for a linear SVM. Possible values are $C \in \{0.05, 0.1, 0.5, 1.0, 5, 10, 1000\}$. Again, the macro-averaged $PRAVG$ on the test set is displayed for each selection strategy. Except for the WebKB data set, all differences are significant and the binomial test also indicates a performance better than random. Again, the $\xi\alpha$-estimator can effectively select a reasonable model. However, as expected, the performance of the leave-one-out method is higher.

Table 6.4 shows how frequently each value of C is selected. Uniformly over almost all categories of the three collections, the $\xi\alpha$-estimator selects $C = 0.5$. While this is a reasonable value, there appears to be a tendency to underestimate the optimal value of C. $C = 1$ is a better choice in terms of the macro-average. This tendency can be explained using the properties of the $\xi\alpha$-estimators identified in Chapter 5. The $\xi\alpha$-estimators have a conservative bias. Therefore they tend to underestimate the true $PRAVG$. The fact that the bias decreases with C (vanishing for the extreme case of $C = 0$) explains the tendency to select a smaller than optimal value of C.

Comparing the average performance of C fixed over all categories given in Table 6.4 with the optimal selection on a per-category basis given in Table 6.4 leads to an interesting observation. A good value of C appears to be a property of the whole collection, rather than of an individual category. For Reuters, keeping $C = 5$ fixed over all categories leads to a near optimal performance. The same is true for WebKB ($C = 1.0$) and Ohsumed ($C = 5.0$). In general, there is a large range of C that leads to good performance.

6.3.3 Non-Linear Classification Rules

An alternative to using a linear SVM and selecting the value of C is capacity adjustment by using non-linear kernels [Vapnik, 1998]. Table 6.5 compares the $\xi\alpha$-estimator and leave-one-out for selecting the kernel width $\gamma \in \{0.01, 0.03, 0.1, 0.3, 1.0, 3.0\}$ of the RBF-kernel (3.34). Again, the $\xi\alpha$-estimator significantly (except WebKB) improves over the random baseline. However, the leave-one-out estimates tend to select better models. While the $\xi\alpha$-estimator leads to a better macro-average on the Reuters tasks, the difference is not significant.

The values of γ selected by the model-selection method are given in Table 6.6. Again, both methods make reasonable selections. However, the $\xi\alpha$-estimator appears to be biased towards smaller values of γ.

RBF-kernel γ		0.01	0.03	0.1	0.3	1.0	3.0
Reuters (top 10)	macroavg. $PRAVG_{test}$	84.5	83.4	83.6	83.8	81.5	58.4
	selection by $PRAVG_{\xi\alpha}$	7	1	1	1	0	0
	selection by $PRAVG_{loo}$	2	3	1	4	0	0
Reuters (all 90)	macroavg. $PRAVG_{test}$	56.7	56.2	56.3	52.6	43.9	25.3
	selection by $PRAVG_{\xi\alpha}$	30	20	16	12	6	6
	selection by $PRAVG_{loo}$	22	20	17	15	10	6
WebKB	macroavg. $PRAVG_{test}$	85.9	85.6	86.2	88.2	87.1	75.8
	selection by $PRAVG_{\xi\alpha}$	2	0	0	2	0	0
	selection by $PRAVG_{loo}$	0	0	1	2	1	0
Ohsumed	macroavg. $PRAVG_{test}$	66.1	66.2	66.6	66.9	60.9	27.9
	selection by $PRAVG_{\xi\alpha}$	7	8	5	3	0	0
	selection by $PRAVG_{loo}$	2	2	6	12	1	0

Table 6.6: Selections of the model-selection criteria for the value of γ in the RBF-kernel. The counts indicate for how many categories each model was selected. The macro-average of $PRAVG_{test}$ illustrates the average model performance.

In general, the average performance is rather insensitive to γ over a large range. Any fixed value of $\gamma \in \{0.01, 0.03, 0.1, 0.3\}$ leads to an average performance close to the optimum given in Table 6.5.

6.3.4 Comparison with Conventional Methods

To get a feel for the relative performance of SVMs and the automatically selected models, the SVM is now compared to four conventional learning methods. These methods are a representative selection from the most popular approaches to learning text classifiers in the inductive setting:

- generative modelling using a naive Bayes classifier with a multinomial mixture model (see Section 2.5.1)
- an adaptation of the Rocchio algorithm as the most popular learning method from information retrieval (see Section 2.5.2)
- C4.5 as the standard decision tree/rule learner (see Section 2.5.4)
- the k-nearest neighbor classifier as a representative of instance-based approaches (see Section 2.5.3)

Each conventional method has shown good results on text-categorization problems in previous studies.

To make sure that the results are not biased in favor of the SVM through an inappropriate experiment design, the conventional methods are given an “unfair advantage”. For the conventional methods tables 6.7 to 6.9 show the PRBEP of the parameter setting that achieves the best performance on the test set. This corresponds to the “optimal” model selection. Therefore, the performance of the conventional methods is optimistically biased. A large number of parameter settings was tried for the conventional methods, namely all combinations of: selecting $N \in \{500, 1000, 2000, 5000, 10000, all\}$ features using mutual information[2], use of stemming, use of stopword removal, binary vs. TFIDF word weight-

[2] With more than 5000 features running C4.5 becomes intractable. The respective experiments are excluded.

					linear SVM		RBF-SVM
	Bayes	Rocchio	C4.5	k-NN	$C = 0.5$	$C = 1.0$	$\gamma = 0.01$
earn	96.0	96.1	96.1	97.8	98.0	98.2	98.1
acq	90.7	92.1	85.3	91.8	95.5	95.6	94.7
money-fx	59.6	67.6	69.4	75.4	78.8	78.5	74.3
grain	69.8	79.5	89.1	82.6	91.9	93.1	93.4
crude	81.2	81.5	75.5	85.8	89.4	89.4	88.7
trade	52.2	77.4	59.2	77.9	79.2	79.2	76.6
interest	57.6	72.5	49.1	76.7	75.6	74.8	69.1
ship	80.9	83.1	80.9	79.8	87.4	86.5	85.8
wheat	63.4	79.4	85.5	72.9	86.6	86.8	82.4
corn	45.2	62.2	87.7	71.4	87.5	87.8	84.6
microavg. (all 90)	72.3	79.9	79.4	82.6	86.7	87.5	86.4

Table 6.7: Precision/recall breakeven point for the Reuters categories comparing the conventional methods with the SVM. The parameters C and γ are chosen as suggested by the model selection criteria. Naive Bayes achieves best performance without stemming and stopword removal using all features. All other conventional methods perform best using 1000 features and stopword removal; C4.5 and Rocchio use also stemming. Furthermore, $\beta = 1.0$ for Rocchio, C4.5 uses TFIDF weighting, and $k = 15$ for k-NN.

ing (only C4.5), $\beta \in \{0, 0.1, 0.25, 0.5, 1.0\}$ (only Rocchio), and $k \in \{1, 15, 30, 45, 60\}$ (only k-NN).

Tables 6.7 to 6.9 show that all SVMs with automatic model selection substantially outperforms all conventional methods. For the SVM, the tables give the performance of the models most frequently selected by the $\xi\alpha$-estimator and by leave-one-out. Not only are the micro-averages of the SVM higher on all three collections, the SVM is also uniformly better on almost all individual categories shown in the tables despite the "unfair advantage" given to the conventional methods. Over all 90 Reuters categories, for example the linear SVM with $C = 0.5$ is better than k-NN on 66 categories (16 ties, 8 worse). This is a significant improvement according to the binomial sign test. The advantage of the SVM over the other conventional methods is even larger. Even more drastically, all SVMs outperforms all conventional methods on all categories of the WebKB collection. Similarly for the Ohsumed data, for example the linear SVM with $C = 0.5$ shows a performance better than the maximum performance of k-NN on all 23 categories.

Finally, how do linear and non-linear SVMs compare among each other? Going beyond the results given in the tables above, even with the optimal value of γ and with an optimally chosen value of C the RBF-SVM does not consistently outperform the linear SVM (with $C = 1.0$). The same is true also for other standard kernels (e.g. polynomials of varying degree). This leads to the conclusion that non-linear SVMs do not provide any advantage for text classification using the standard kernels. Furthermore, the non-linear kernels take substantially longer to train. However, specially designed kernels introducing new prior knowledge about the classification task may provide benefits.

Recently, Yang and Liu [Yang and Liu, 1999] also compared SVMs to other text-classification methods on a modified version of the Reuters collection. While in their experiments SVMs also turned out to be the best performer with $F_1 = 85.99\%$ (micro-averaged), the advantage over k-NN with $F_1 = 85.67\%$ (micro-averaged) was only marginal. This outcome is due to several problems in Yang and Liu's experimental setup. Most importantly, although the performance is measured in terms of F_1, only k-NN is optimized to maximize F_1, while the SVM is optimized for error rate. This makes the SVM sacrifice

					linear SVM		RBF-SVM	
	Bayes	Rocchio	C4.5	k-NN	$C = 0.5$	$C = 1.0$	$\gamma = 0.01$	$\gamma = 0.3$
course	93.2	95.5	83.0	90.9	96.1	95.9	95.5	96.1
faculty	74.5	80.0	73.8	62.4	90.1	88.2	85.3	85.3
project	55.0	42.0	47.0	55.0	69.3	75.0	80.0	76.8
student	90.6	85.2	85.1	90.6	93.0	93.0	92.2	92.5
microavg.	82.0	74.1	79.1	80.5	90.3	91.6	89.4	90.1
Err_{test}	13.7	23.9	19.9	21.2	7.5	8.0	9.7	9.3

Table 6.8: PRBEP and classification accuracy for the WebKB categories comparing the conventional methods with the SVM. All conventional methods do not use stemming. Naive Bayes and Rocchio perform best using stopword removal and 500 features. C4.5 also prefers 500 (binary) features, but no stopword removal. k-NN uses 2000 features and $k = 60$. The optimum β for Rocchio is 1.0.

					linear SVM		RBF-SVM	
	Bayes	Rocchio	C4.5	k-NN	$C = 0.5$	$C = 1.0$	$\gamma = 0.03$	$\gamma = 0.3$
Pathology	53.6	52.3	48.7	54.4	59.6	58.6	52.1	55.2
Cardiovascular	74.7	67.2	69.1	75.1	80.2	80.2	75.3	77.8
Neoplasms	80.8	78.7	78.8	78.8	86.3	85.5	82.1	84.4
Nervous System	64.5	64.8	49.0	65.2	72.2	72.5	69.1	70.8
Immunologic	62.6	59.1	66.9	67.0	75.4	74.8	70.3	73.3
microavg. (all 23)	62.4	61.5	56.7	63.4	71.6	71.5	67.7	69.8

Table 6.9: PRBEP for the Ohsumed categories comparing the conventional methods with the SVM. Naive Bayes and k-NN ($k = 45$) perform best using all features after stemming and stopword removal. The peak performance of Rocchio is for 5000 features without stemming and stopword removal for $\beta = 1.0$. For C4.5 the best performance is achieved for 2000 features with TFIDF weighting and stemming. More than 2000 features are computationally intractable for C4.5 on this task due to time and memory constraints.

recall for precision, leading to a lower F_1 score. A more suitable approach is pursued in the following. An SVM with cost model (see Section 3.4) and a cost ratio proportional to the fraction of negative to positive examples is used to boost recall. For $C = 0.5$ on the same variant of the Reuters collection as used in [Yang and Liu, 1999] this leads to a micro-averaged F_1 of 88.04%. This is again a substantial gain over the k-NN classifier. Previous comparisons [Joachims, 1997b][Dumais et al., 1998] came to similar conclusions.

In terms of computational efficiency, linear SVMs are very fast both at training and at classification. In all experiments SVM^{light} was used as the training algorithm. On average, it took 1.3 seconds on a Pentium III/500Mhz (with 128MB of a RAM running Linux) to train a single Reuters category for $C = 0.5$. Classifying a new document vector is essentially equivalent to computing a single (sparse) dot product. SVMs with non-linear kernels take substantially more time to train and to apply. The average training time for an RBF-kernel with $\gamma = 0.1$ is 38.1 seconds on the 90 Reuters categories. More details are given in Chapter 8.

6.4 Related Work

For the Reuters collection, the experimental results originally published in [Joachims, 1997b][Joachims, 1998b] were later verified in [Dumais et al., 1998]. Dumais et al. used a different document representation, but came to a similar performance as reported here. They used a binary representation with feature selection by empirical mutual information. Further experimental results using SVMs for text classification can be found in [Taira and Haruno, 1999] [Drucker et al., 1999] [Yang and Liu, 1999] [Neumann and Schmeier, 1999] [Busemann et al., 2000] [Kindermann et al., 2000].

As already noted, other margin-based methods like boosted decisions stumps show excellent results for text classification as well [Schapire et al., 1998] [Schapire and Singer, 2000]. Schapire et al. report a performance that is only slightly worse than for the SVM on the Reuters collection. However, training is computationally less efficient. They report that it took three days of CPU-time to train on the Reuters corpus. However, it is likely that training can be sped up. An interesting open question is a theoretical analysis according to Chapter 4.

The best result on the Reuters data reported in the literature is a precision/recall breakeven point of 87.8 (compared to 87.5 here) [Weiss et al., 1999]. This result is for boosting 100 decision trees per category. Weiss et al. do not discuss computational efficiency, but training such large models must take substantially longer than training a linear SVM. Training a single C4.5 decision tree already takes an order of magnitude longer than training a linear SVM. Furthermore, their approach involves feature selection. This adds a processing step with additional parameters and implicit assumptions, which makes training more complicated. From a theoretical perspective, such complex models with greedy optimization steps in the learning phase are not well understood. There is less guidance in how and when this approach is appropriate. In particular, efficient model selection criteria, like they are discussed in this chapter for SVMs, are likely to not exist.

6.5 Summary and Conclusions

This chapter evaluated the performance and practicability of inductive SVMs for learning text classifiers. It recognized that solving a learning task is not restricted to simply training the learner, but that preprocessing steps like choosing an appropriate representation are equally important.

Therefore, this chapter proposed and evaluated approaches to model selection for support vector machines in the context of text classification. It compared a new model-selection strategy based on $\xi\alpha$-estimates with leave-one-out estimation. The new method is computationally much more efficient than cross-validation or bootstrap. After only a single training run per model, it can effectively select a reasonable model. While the new method provides a large advantage in terms of computational efficiency, the quality of the selected model is better for the exact leave-one-out estimate. This implies that the new method is best suited in situations where response time or computational resources are limited.

This chapter also provides experimental evidence regarding the generalization performance of SVMs for text classification compared to conventional methods. The experimental results show that SVMs achieve excellent performance on three different text-classification tasks, outperforming conventional methods substantially. SVMs eliminate the need for feature selection, saving a complicated preprocessing step. In general, SVMs are fairly insensitive to particular parameter settings over a large range. For all parameters like the use of stemming, stopword removal, or kernels, the new model-selection strategy can efficiently avoid bad parameter values. This makes SVMs easy to use and apply in real-world applications where expert interventions are not possible.

Chapter 7

Transductive Text Classification

For many practical uses of text classification, it is crucial that the learner be able to generalize well using little training data. A news-filtering service, for example, requiring a hundred days' worth of training data is unlikely to please even the most patient users. The work presented in the following tackles the problem of learning from small training samples by taking a *transductive* [Vapnik, 1998], instead of an inductive approach. In the inductive setting the learner tries to induce a decision function which has a low error rate on the whole distribution of examples for the particular learning task. Often, this setting is unnecessarily complex. In many situations we do not care about the particular decision function, but rather that we classify a *given set of examples (i.e. a test set) with as few errors as possible.* This is the goal of transductive inference.

Some examples of transductive text-classification tasks are the following. All have in common that there is little training data, but a very large test set.

RELEVANCE FEEDBACK: This is a standard technique in free-text information retrieval. The user marks some documents returned by an initial query as relevant or irrelevant. These compose the training set of a text-classification task, while the remaining document database is the test set. The user is interested in a good classification of the test set into those documents relevant or irrelevant to the query.

NETNEWS FILTERING: Each day a large number of netnews articles is posted. Given the few training examples the user labeled on previous days, he or she wants today's most interesting articles.

REORGANIZING A DOCUMENT COLLECTION: With the advance of paperless offices, companies have started using document databases with classification schemes. When introducing new categories, they need text classifiers which, given some training examples, classify the rest of the database automatically.

This chapter introduces *Transductive Support Vector Machines* (TSVMs) to text classification. They substantially improve the already excellent performance of SVMs for text classification. Especially for very small training sets, TSVMs reduce the required amount of labeled training data down to a twentieth for some tasks. A new algorithm that facilitates the large-scale transductive learning needed for text classification is proposed in Chapter 9. It can efficiently train TSVMs with 10,000 examples and more.

7.1 Learning Task

The setting of transductive inference was introduced by Vapnik (see for example [Vapnik, 1998]). For a learning task $\Pr(\vec{x}, y) = \Pr(y|\vec{x}) \Pr(\vec{x})$ the learner $\mathcal{L}$ is given a hypothesis space $\mathcal{H}$ of functions $h : X \longrightarrow \{-1, 1\}$ and an i.i.d. sample S_{train} of n training examples

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_n, y_n) \tag{7.1}$$

Each training example consists of a document vector $\vec{x} \in X$ and a binary label $y \in \{-1, +1\}$. In contrast to the inductive setting, the learner is also given an i.i.d. sample S_{test} of k test examples

$$\vec{x}_1^*, \vec{x}_2^*, ..., \vec{x}_k^* \tag{7.2}$$

from the same distribution. The transductive learner L aims to selects a function $h_{\mathcal{L}} = \mathcal{L}(S_{train}, S_{test})$ from $\mathcal{H}$ using S_{train} and S_{test} so that the expected number of erroneous predictions

$$Err^{n,k}(\mathcal{L}) = \int \frac{1}{k} \sum_{i=1}^{k} L_{0/1}(h_{\mathcal{L}}(\vec{x}_i^*), y_i^*) d\Pr(\vec{x}_1, y_1) \cdots d\Pr(\vec{x}_k^*, y_k^*)$$

on the test examples is minimized. $L_{0/1}(a, b)$ is zero if $a = b$, otherwise it is one. Vapnik [Vapnik, 1998] gives bounds on the relative uniform deviation of training error

$$Err_{emp}(h) = \frac{1}{n} \sum_{i=1}^{n} L_{0/1}(h(\vec{x}_i), y_i) \tag{7.3}$$

and test error

$$Err_{test}(h) = \frac{1}{k} \sum_{i=1}^{k} L_{0/1}(h(\vec{x}_i^*), y_i^{true}). \tag{7.4}$$

With probability $1 - \eta$

$$Err_{test}(h) \leq Err_{emp}(h) + \Omega(n, k, d, \eta) \tag{7.5}$$

where the confidence interval $\Omega(n, k, d, \eta)$ depends on the number of training examples n, the number of test examples k, and the VC-dimension d of $\mathcal{H}$ (see [Vapnik, 1998] for details).

This problem of transductive inference may not seem profoundly different from the usual inductive setting studied in machine learning. One could learn a classification rule based on the training data and then apply it to the test data afterwards. Nevertheless, to solve the problem of estimating k binary values $y_1^*, ..., y_k^*$ we need to solve the more complex problem of estimating a function over a possibly continuous space. This may not be the best solution when the size n of the training sample (7.1) is small.

What information do we get from studying the test sample (7.2) and how can we use it? The training and the test sample split the hypothesis space $\mathcal{H}$ into a finite number of equivalence classes $\mathcal{H}'$. Two functions from $\mathcal{H}$ belong to the same equivalence class if they both classify the training and the test sample in the same way. This reduces the learning problem from finding a function in the possibly infinite set $\mathcal{H}$ to finding one of finitely many equivalence classes $\mathcal{H}'$. Most importantly, we can use these equivalence classes to build a structure of increasing VC-dimension for *structural risk minimization* [Vapnik, 1998].

$$\mathcal{H}'_1 \subset \mathcal{H}'_2 \subset \cdots \subset \mathcal{H}' \tag{7.6}$$

Unlike in the inductive setting, we can study the location of the test examples when defining the structure. Using prior knowledge about the nature of $\Pr(\vec{x}, y)$ we can build a more appropriate structure and learn more quickly. What this means for text classification is analyzed in Section 7.3.

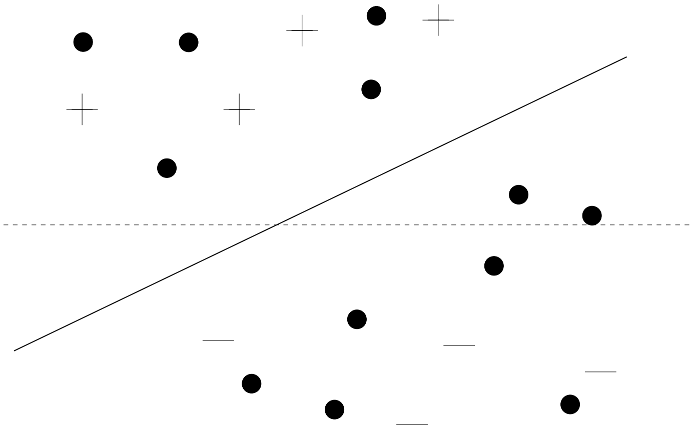


Figure 7.1: The maximum-margin hyperplanes. Positive/negative examples are marked as +/−, test examples as dots. The dashed line is the solution of the inductive SVM. The solid line shows the transductive classification.

7.2 Transductive Support Vector Machines

For support vector machines, we can build the structure of increasing VC-dimension based on the margin of separating hyperplanes on both the training and the test data. Vapnik shows that with the size of the margin we can control the maximum number of equivalence classes (i. e. the VC-dimension).

Theorem 12 ([Vapnik, 1998]) *Consider hyperplanes $h(\vec{x}) = sign\{\vec{x} \cdot \vec{w} + b\}$ as hypothesis space $\mathcal{H}$. If the attribute vectors of a training sample (7.1) and a test sample (7.2) are contained in a ball of diameter R, then there are at most*

$$N_r < exp\left(d\left(\frac{n+k}{d}+1\right)\right), d = min\left(N, \left[\frac{R^2}{\delta^2}\right]+1\right)$$

equivalence classes which contain a separating hyperplane with

$$\forall_{i=1}^{n} \left|\frac{\vec{w}}{||\vec{w}||} \cdot \vec{x}_i + b\right| \geq \delta \qquad \forall_{j=1}^{k} \left|\frac{\vec{w}}{||\vec{w}||} \cdot \vec{x}_j^* + b\right| \geq \delta$$

(i.e. margin larger or equal to δ). N is the dimensionality of the space, and $[c]$ is the integer part of c.

Note that the VC-dimension does not necessarily depend on the number of features, but can be much lower than the dimensionality of the space. Let us use this structure based on the margin of separating hyperplanes. Structural risk minimization tells us that we get the smallest bound on the test error if we select the equivalence class from the structure element $\mathcal{H}_i'$ which minimizes (7.5). For linearly separable problems this leads to the following optimization problem [Vapnik, 1998].

Optimization Problem 9 (Transductive SVM (linearly separable case))

$$\begin{array}{ll} \textit{minimize:} & V(y_1^*, ..., y_k^*, \vec{w}, b) = \frac{1}{2}\vec{w} \cdot \vec{w} \\ \textit{subject to:} & \forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 \\ & \forall_{j=1}^{k} : y_j^*[\vec{w} \cdot \vec{x}_j^* + b] \geq 1 \\ & \forall_{j=1}^{k} : y_j^* \in \{-1, +1\} \end{array}$$

Solving this problem means finding a labelling $y_1^*, ..., y_k^*$ of the test data and a hyperplane $< \vec{w}, b >$, so that this hyperplane separates both training and test data with maximum margin. Figure 7.1 illustrates this. To be able to handle non-separable data, we can introduce slack variables ξ_i similarly to the way we do with inductive SVMs.

Optimization Problem 10 (Transductive SVM (non-separable case))

$$\begin{array}{ll}
\textit{minimize:} & W(y_1^*, ..., y_k^*, \vec{w}, b, \xi_1, ..., \xi_n, \xi_1^*, ..., \xi_k^*) = \frac{1}{2}\vec{w} \cdot \vec{w} + C \sum_{i=0}^{n} \xi_i + C^* \sum_{j=0}^{k} \xi_j^* \\
\textit{subject to:} & \forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i \\
 & \forall_{j=1}^{k} : y_j^*[\vec{w} \cdot \vec{x}_j^* + b] \geq 1 - \xi_j^* \\
 & \forall_{i=1}^{n} : \xi_i > 0 \\
 & \forall_{j=1}^{k} : \xi_j^* > 0 \\
 & \forall_{j=1}^{k} : y_j^* \in \{-1, +1\}
\end{array}$$

C and C^* are parameters set by the user. They allow trading off margin size against misclassifying training examples or excluding test examples. C^* can be used reduce sensitivity towards outliers (i.e. single examples falsely reducing the margin on the test data). How this optimization problem can be solved efficiently is the subject of Chapter 9.

7.3 What Makes TSVMs Especially well Suited for Text Classification?

In Chapter 4 text-classification tasks are characterized by a special set of properties. It is shown how these properties lead to good learning results for an inductive SVM. TSVMs inherit most properties of inductive SVMs. Since the argumentation was based on the margin of text-classification tasks, most arguments apply to TSVMs as well. But how can TSVMs be any better?

7.3.1 An Intuitive Example

In the field of information retrieval it is well known that words in natural language occur in strong co-occurrence patterns (see e.g. [van Rijsbergen, 1977]). Some words are likely to occur together in one document, others are not. For examples, when asking the search engine Altavista about all documents containing the words `pepper` and `salt`, it returns 327,180 web pages. When asking for the documents with the words `pepper` and `physics`, we get only 4,220 hits, although `physics` is a more popular word on the web than `salt`. Many approaches in information retrieval try to exploit this cluster structure of text (see e.g. [Baeza-Yates and Ribeiro-Neto, 1999, Chapter 5]). And it is this co-occurrence information that TSVMs exploit as prior knowledge about the learning task.

Consider the example in Figure 7.2. Imagine document $D1$ was given as a training example for class A and document $D6$ was given as a training example for class B. How should we classify documents $D2$ to $D4$ (the test set)? Even if we did not understand the meaning of the words, we would classify $D2$ and $D3$ into class A, and $D3$ and $D4$ into class B. We would do so even though $D1$ and $D3$ do not share any informative words. The reason we choose this classification of the test data over the others stems from our prior knowledge about the properties of text and common text-classification tasks. Often we want to classify documents by topic, source, or style. For these type of

	nuclear	physics	atom	parsley	basil	salt	and
D1	**1**						**1**
D2	1	1	1				1
D3			1				1
D4				1	1		1
D5				1		1	1
D6					**1**	**1**	**1**

Figure 7.2: Example of a text-classification problem with co-occurrence pattern. Rows correspond to documents, columns to words. A table entry of 1 denotes the occurrence of a word in a document.

classification tasks we find stronger co-occurrence patterns within categories than between different categories. In our example we analyzed the co-occurrence information in the test data and found two clusters. These clusters indicate different topics of $\{D1, D2, D3\}$ vs. $\{D4, D5, D6\}$, and we choose the cluster separator as our classification. Note again that we got to this classification by studying the location of the test examples, which is not possible for an inductive learner.

The TSVM outputs the same classification as we suggested above, although all 16 dichotomies of $D2$ to $D5$ can be achieved with linear separators. Assigning $D2$ and $D3$ to class A and $D3$ and $D4$ to class B is the maximum-margin solution (i.e. the solution of Optimization Problem 9). We see that the maximum-margin bias reflects our prior knowledge about text classification well. By analyzing the test set, we can exploit this prior knowledge for learning.

In this example taking a transductive approach is obviously beneficial. Is this just an exceptional constellation or does the same effect occur on a more general class of problems?

7.3.2 Transductive Learning of TCat-Concepts

The following simulation study analyzes TCat-concepts (see Chapter 4) for transductive learning. It shows how a TSVM takes advantage of the test set and explains why TSVMs can learn TCat-concepts more efficiently than inductive SVMs, if sufficient test data is available.

For simplicity reasons, consider the small concept

$$TCat([1:0:1],[0:1:1],[4:4:8]). \tag{7.7}$$

It includes 10 features of which 8 features have no connection with the class label. The remaining two features indicate the positive and the negative class respectively. Each document contains 5 words. The distribution $\Pr(X)$ of feature vectors is such that each binary vector consistent with the TCat-concept is equally likely.

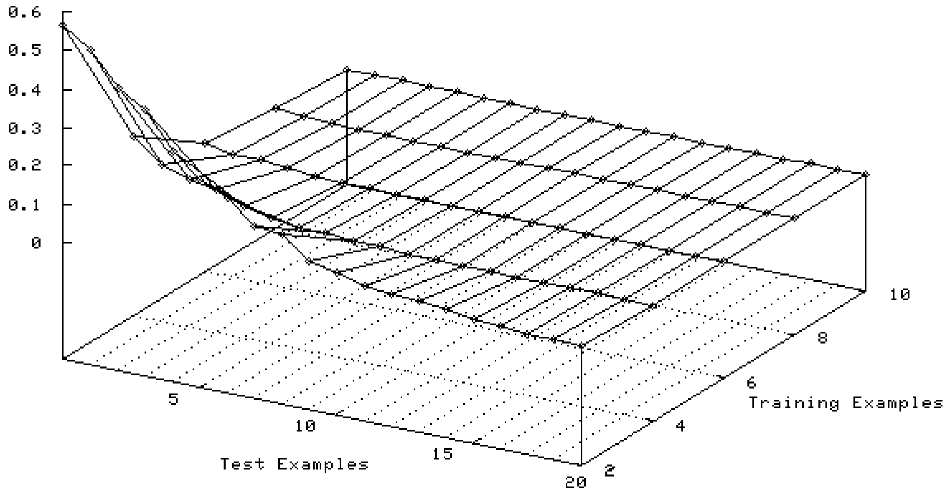


Figure 7.3: Simulation of a transductive SVM on a TCat-concept.

Figure 7.3 contains the learning curve of a TSVM averaged over multiple training and test sets. The vertical axis shows the generalization error depending on the size of the training and the test set. Not suprisingly, the prediction error on the test set decreases rapidly with more training examples. But the figure also shows that the prediction error goes down for an increasing test set size. Why does this happen?

Figure 7.4 illustrates how the TSVM behaves for various test set sizes. Plus signs indicate positive examples, minus sign indicate negative examples, and dots stand for test examples. In a) the test set is not used and the hyperplane indicated by the dashed line is identical to that of an inductive SVM. After "adding" some test examples in b), it turns out that the dashed line does not really have a large margin. It just happens to have a large margin for the particular training sample.

In fact, the dashed line has a smaller margin than the correct labeling of the test examples. Since the data is consistent with the TCat-concept from (7.7), it is possible to compute that the correct labeling of the test examples induces a margin of at least $\delta \geq \frac{1}{\sqrt{2}}$. This was shown in Lemma 2 from Chapter 4. Given the test data in b) only two splits of the test examples have a sufficiently large margin. They are indicated by the shaded regions in b). All other separations can be excluded from consideration, since they violate the margin restriction and therefore cannot be the correct labeling of the test data.

After adding even more test data in c), only one hyperplane is both consistent with the training examples and has a sufficiently large margin. Clearly, this hyperplane represents the labeling of the test examples produced by the TSVM. It makes fewer errors on the test set than the labeling of an inductive SVM.

This machanism is not limited to small TCat-concepts like in (7.7). It applies also to more complicated TCat-concepts. The benefit of transductive over inductive learning depends on two factors. While the size of the margin is important, even more important is $\Pr(X)$. In the simulation $\Pr(X)$ was chosen to be uniform everywhere but within the margin. Feature vectors lying inside the margin have probabilty zero. This is the "best case". For sufficiently large test sets, there are test examples almost everywhere but within the margin. So the test examples effectively rule out all separations but the correct one. For other $\Pr(X)$ that have zero probability in some regions outside the margin or a small

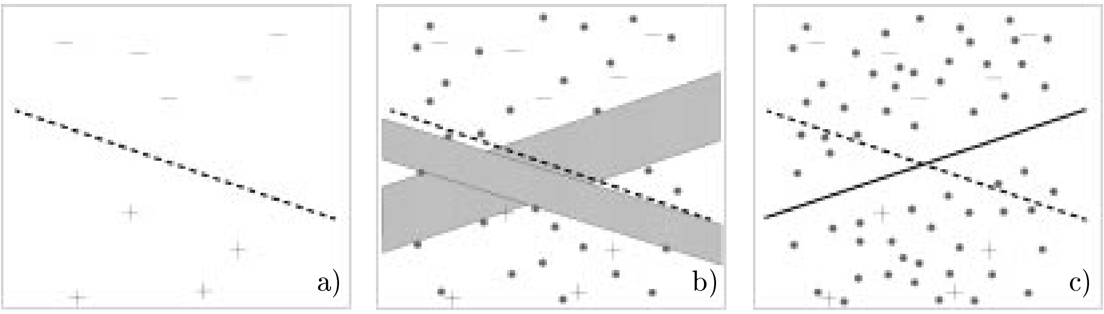


Figure 7.4: Example of a transductive SVM exploiting the location of the test examples.

probability of examples within the margin, the benefit of transductive learning can be smaller.

This simulation study gives a qualitative explanation for the benefit of transductive SVMs over inductive SVMs. It should also be possible to prove quantitative bounds connecting error and test set size. Nevertheless, these results will be of little practical use, because such bounds must depend on $\Pr(X)$. Since it is not practical to estimate $\Pr(X)$ for real-world tasks, finding bounds for particular $\Pr(X)$ is left as an open question. Instead, the quantitative benefit of a transductive approach to text classification is evaluated in the following experiments.

7.4 Experiments

The empirical evaluation is done on three test collection (see Section 2.7.1). Unless noted otherwise, the evaluation follows the setup described in Section 2.7.

The first collection is the Reuters-21578 dataset. The "ModApte" split is used, leading to a corpus of 9,603 training documents and 3,299 test documents. Of the 135 potential topic categories, only the most frequent 10 are used, while keeping all documents. Both stemming and stopword removal are performed.

The second dataset is the WebKB collection. Following the setup in [Nigam et al., 1998], only the classes `course`, `faculty`, `project`, and `student` are used. Documents not in one of these classes are deleted. After removing documents which just contain the relocation command for the browser, this leaves 4,183 examples. The pages from Cornell University are used for training, while all other pages are used for testing. Like in [Nigam et al., 1998], stemming and stopword removal are not used.

The third test collection is taken from the Ohsumed corpus. From the 50,216 documents in 1991 which have abstracts, the first 10,000 are used for training and the second 10,000 are used for testing. The task is to assign documents to one or multiple categories of the 5 most frequent MeSH "diseases" categories. Both stemming and stop-word removal are performed.

The following experiments show the effect of using the transductive SVM instead of inductive methods. To provide a baseline for comparison, the results of the inductive SVM and a multinomial naive Bayes classifier as described in Section 2.5.1 are added. Where applicable, the results are averaged over a number of random training (test) samples.

Figure 7.5 gives the results for the Reuters dataset. For training sets of 17 documents and test sets of 3,299 documents, the transductive SVM leads to an improved performance on all categories, raising the macro-average of the precision/recall breakeven points from

	Bayes	SVM	TSVM
earn	78.8	91.3	95.4
acq	57.4	67.8	76.6
money-fx	43.9	41.3	60.0
grain	40.1	56.2	68.5
crude	24.8	40.9	83.6
trade	22.1	29.5	34.0
interest	24.5	35.6	50.8
ship	33.2	32.5	46.3
wheat	19.5	47.9	54.4
corn	14.5	41.3	43.7
macro-average	35.9	48.4	60.8

Figure 7.5: Average precision/recall breakeven points for the ten most frequent Reuters categories using 17 training and 3,299 test examples. Naive Bayes uses feature selection by empirical mutual information with dictionaries of size 1,000. No feature selection was done for SVM and TSVM.

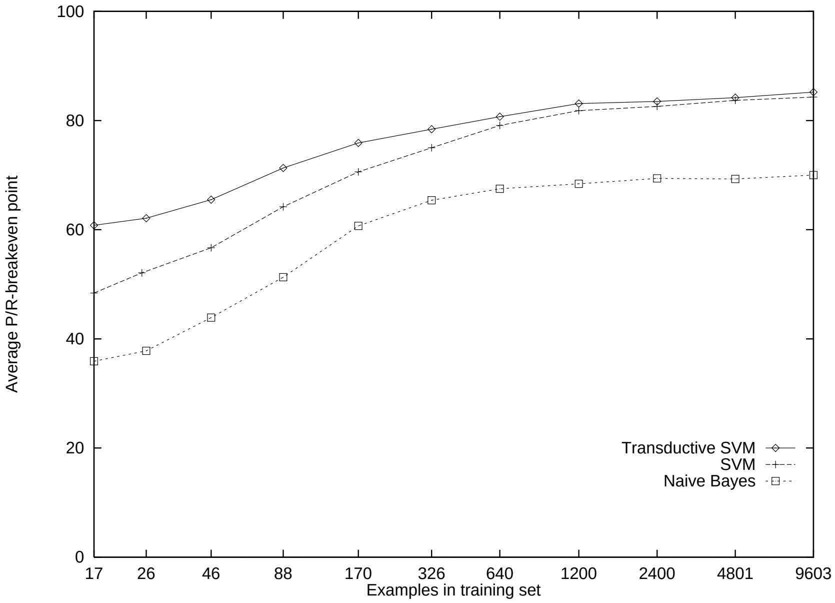


Figure 7.6: Macro-averaged PRBEP on the Reuters dataset for different training set sizes and a test set size of 3,299.

48.4 for the inductive SVM to 60.8. These averages correspond to the left-most points in Figure 7.6. This graph shows the effect of varying the size of the training set. The advantage of using the transductive approach is largest for small training sets. For increasing training set size, the performance of the SVM approaches that of the TSVM. The influence of the test set size on the performance of the TSVM is displayed in Figure 7.7. The bigger the test set, the larger the performance gap between SVM and TSVM. Adding more test examples beyond 3,299 is not likely to increase performance by much, since the graph is already very flat.

The results on the WebKB dataset are similar (Figure 7.8). The average of the

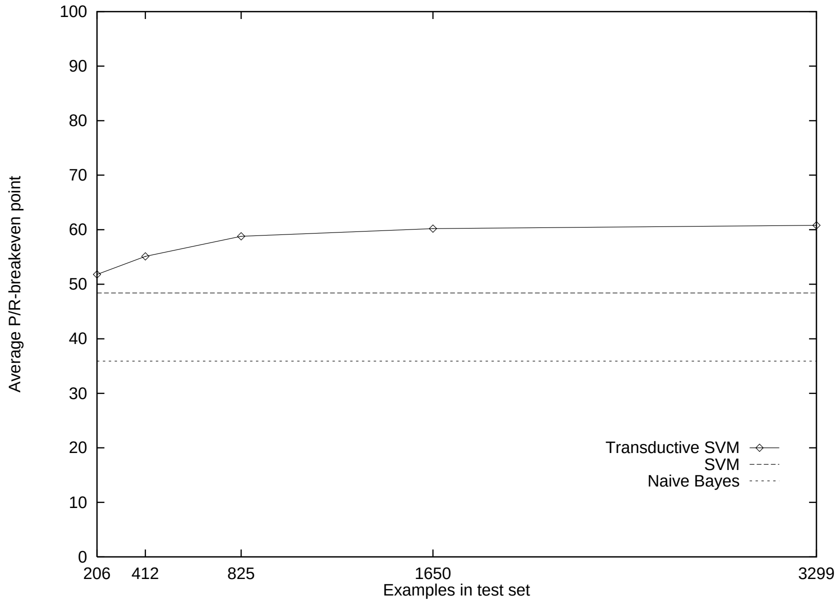


Figure 7.7: Macro-averaged PRBEP on the Reuters dataset for 17 training documents and varying test set size for the TSVM.

	Bayes	SVM	TSVM
course	57.2	68.7	93.8
faculty	42.4	52.5	53.7
project	21.4	37.5	18.4
student	63.5	70.0	83.8
macro-average	46.1	57.2	62.4

Figure 7.8: Average precision/recall breakeven points for the WebKB categories using 9 training and 3957 test examples. Naive Bayes uses a dictionary with the 2,000 highest mutual information words. No feature selection was done for the SVM. Due to the large number of words, the TSVM used only those words which occur at least 5 times in the whole sample.

P/R-breakeven points increases from 57.2 to 62.4 by using the transductive approach. Nevertheless, for the category `project` the TSVM performs substantially worse, while the gain on the category `course` is large. Let us look at this in more detail. Figures 7.9 and 7.10 show how the performance changes with increasing training set size for `course` and `project`. While for `course` the TSVM nearly reaches its peak performance immediately, it needs more training examples to surpass the inductive SVM for `project`. Why does this happen?

First, `project` is the least populous class. Among 9 training examples, there is only one from the `project` category. But more importantly, a look at the project pages reveals that many of them give a description of the project topic. My conjecture is that the margin along this “topic dimension” is large, and so the TSVM tries to separate the test data by topic. Only when there are enough project pages with different topics in the training set is the generalization along the project topic ruled out. Most course pages at Cornell, on

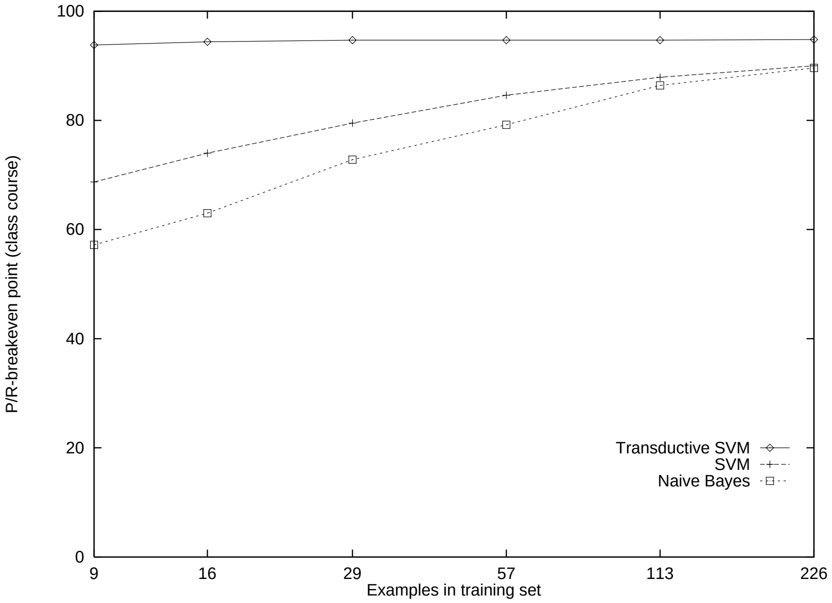


Figure 7.9: Average PRBEP on the WebKB category `course` for different training set sizes.

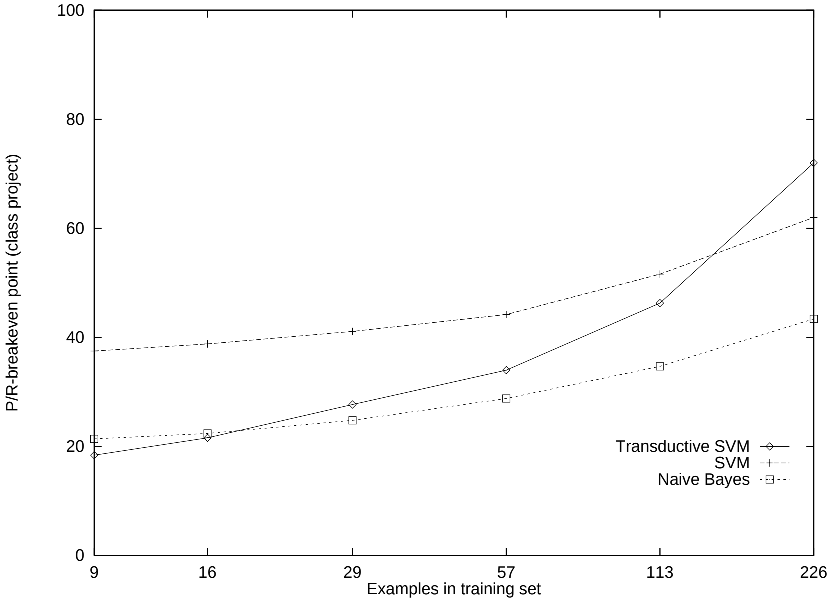


Figure 7.10: Average PRBEP on the WebKB category `project` for different training set sizes.

	Bayes	SVM	TSVM
Pathology	39.6	41.8	43.4
Cardiovascular	49.0	58.0	69.1
Neoplasms	53.1	65.1	70.3
Nervous System	28.1	35.5	38.1
Immunologic	28.3	42.8	46.7
macro-average	39.6	48.6	53.5

Figure 7.11: Average PRBEP for the Ohsumed categories using 120 training and 10,000 test examples. Here, Naive Bayes uses dictionaries of 1,000 words selected by mutual information. No feature selection was done for the SVM. The TSVM again uses all words that occur at least 5 times in the whole sample.

the other hand, do not give much topic information besides the title, but rather link to assignments, lecture notes etc. So the TSVM is not “distracted” by large margins along the topics.

The results in Figure 7.11 for the Ohsumed collection complete the empirical evidence given in this section, also supporting its point.

7.5 Constraints on the Transductive Hyperplane

While the transductive approach is beneficial for most tasks, it can also lead to worse performance like for the WebKB category `project`. Is it possible to detect when the TSVM performs well and when it fails?

If additional labeled data is available, it can be used to get an estimate of the prediction error using cross-validation or hold-out testing. Nevertheless, labeled data is usually very scarce. The following presents a criterion that does not require labeled data. It can be computed immediately after running the TSVM at essentially no additional computational expense. It acts as an alarm signal for some cases in which the transductive SVMs fails.

The criterion is based on the observation that the following quantities have the same expected value.

Corollary 3 (Characterization of Optimal TSVM Solution) *Given that the transductive SVM splits the data according to the true decision surface, the following equalities hold:*

$$\mathcal{E}\left(\frac{|\{i:(\alpha_i R_\Delta^2+\xi_i)\geq 1\}|}{n}\right)=\mathcal{E}\left(\frac{|\{i:(\alpha_i R_\Delta^2+\xi_i)\geq 1\}|}{k}\right) \tag{7.8}$$

$$\mathcal{E}\left(\frac{|\{i:y_i=1\wedge(\alpha_i R_\Delta^2+\xi_i)\geq 1\}|}{|\{i:y_i=1\}|}\right)=\mathcal{E}\left(\frac{|\{i:y_i^*=1\wedge(\alpha_i^* R_\Delta^2+\xi_i^*)\geq 1\}|}{|\{i:y_i^*=1\}|}\right) \tag{7.9}$$

$$\mathcal{E}\left(\frac{|\{i:y_i=-1\wedge(\alpha_i R_\Delta^2+\xi_i)\geq 1\}|}{|\{i:y_i=-1\}|}\right)=\mathcal{E}\left(\frac{|\{i:y_i^*=-1\wedge(\alpha_i^* R_\Delta^2+\xi_i^*)\geq 1\}|}{|\{i:y_i^*=-1\}|}\right) \tag{7.10}$$

The expectations are over training/test sets.

Proof *Follows directly from the fact that the examples are i.i.d. However, it needs to be ensured that $\vec{\alpha}$ is selected at random, if the solution of the dual is not unique.* ▮

What is the intuition behind this corollary? Given training sets $(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)$ and test sets $\vec{x}_1^*, ..., \vec{x}_k^*$, the TSVM outputs labelings $y_1^*, ..., y_k^*$ as well as values α_i (or α_i^*) and ξ_i (or ξ_i^*) for each training (or test) example. The corollary shows that if the TSVM labels the test data correctly, then the α and ξ will on average behave the same for both training and test examples. In particular, it states that quantities related to the $\xi\alpha$-estimators from Chapter 5 have the same expected values on both the training and the test samples.

The corollary can be used as follows. If for the current training and test sample the empirical average on the left-hand side is very different from the empirical average on the right-hand side, it is unlikely that the TSVM found the correct labeling of the test data.

For example, suppose that the TSVM finds a labeling of the test examples with $\xi_i = \xi_j^* = 0$ and α-values

train:	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}
	10	12	8	3	100	22	0	134	14	2

test:	α_1^*	α_2^*	α_3^*	α_4^*	α_5^*	α_6^*	α_7^*	α_8^*	α_9^*	α_{10}^*	α_{11}^*	α_{12}^*	α_{13}^*	α_{14}^*	α_{15}^*
	0	0	17	0	0	0	340	0	0	0	143	0	0	0	0

Consider the criterion in (7.8). For $R_\Delta^2 = 1$ this implies that there are $f_{train} = 9$ out of 10 training examples fulfilling $(\alpha_i R_\Delta^2 + \xi_i) \geq 1$ while only $f_{test} = 3$ out of 15 test examples fulfill $(\alpha_i^* R_\Delta^2 + \xi_i) \geq 1$. If the TSVM did find the correct labeling of the test examples, then this outcome of α-values would be unlikely. A hypothesis test against the hypergeometric distribution rejects (7.8) with confidence

$$\Pr\left(\left|\frac{f_{train}}{10} - \frac{f_{test}}{15}\right| \geq \frac{7}{10}\right) \leq 0.01 \tag{7.11}$$

This indicates that the TSVM probably did not find the correct labeling.

Furthermore, the α-values have an intuitive meaning in and of themselves. It is related to the $\xi\alpha$-estimators from Chapter 5. Keeping the labeling of the test examples fixed, $\frac{f_{train}}{n}$ and $\frac{f_{test}}{k}$ are the fractions of training and test examples that can produce a leave-one-out error. So for the example above, most training examples are "outliers". The training examples will likely lead to errors in leave-one-out testing for an inductive SVM on the whole sample. While this indicates that the solution found by the TSVM is rather strange, note that $\xi\alpha$-estimators are valid estimators only for inductive SVMs. However, this relationship points to an interesting view on transduction as labeling the test examples to minimize leave-one-out error.

Figure 7.12 shows that the criterion in (7.8) can detect the failure of the TSVM on the WebKB category `project`. The figure plots $\frac{f_{train}}{n}$ and $\frac{f_{test}}{k}$ for each WebKB category. For `course`, `faculty`, and `student` both values are relatively close together. Only for small training set sizes of the `project`-category the two values are substantially different. These are exactly the cases where the TSVM shows bad generalization performance.

7.6 Relation to Other Approaches Using Unlabeled Data

This section discusses how transductive learning compares to other approaches that make use of unlabeled data. These are in particular expectation/maximization with the test labels as latent variables, co-training, and other work on transduction.

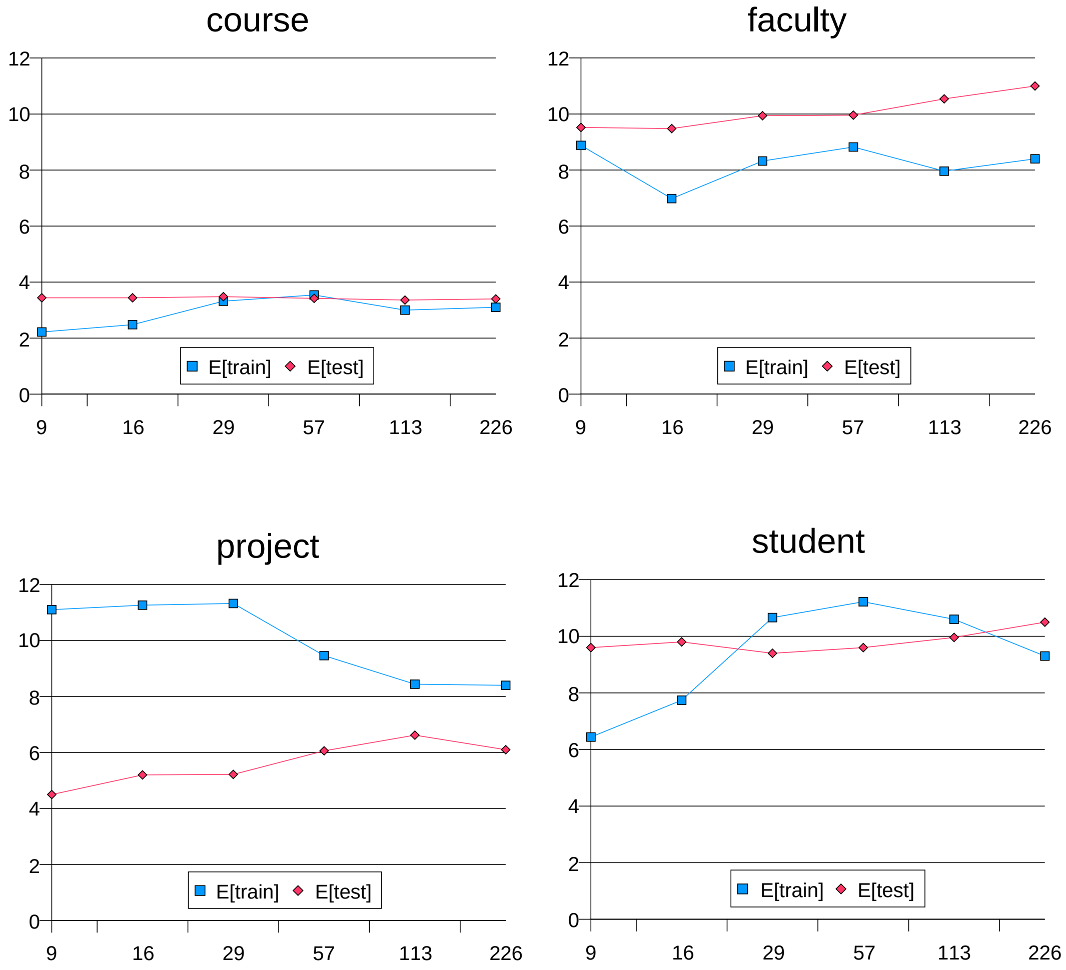


Figure 7.12: Properties of the TSVM solution for the WebKB data set.

7.6.1 Probabilistic Approaches using EM

Previously, Nigam et al. [Nigam et al., 1998][Nigam et al., 2000] proposed another approach to using unlabeled data for text classification. Their success has motivated the work presented here. They use a multinomial Naive Bayes classifier and incorporate unlabeled data using the EM-algorithm. One problem with using Naive Bayes is that its independence assumption is clearly violated for text. Nevertheless, using EM showed substantial improvements over the performance of a regular Naive Bayes classifier.

There are two differences between the approach of Nigam et al. and the work presented here. First, unlabeled data is not expected to be the test set like in transduction. While each approach can probably be extended to both cases, a more fundamental difference lies in the proposed learning method. While Nigam et al. fit a generative model to the unlabeled data, the TSVM takes a discriminative approach. It does not consider a generative model for $\Pr(X)$, but uses the unlabeled data directly for finding low density regions in $\Pr(X)$. The advantage of not having to design a generative model for natural language was already outlined in Chapter 4.

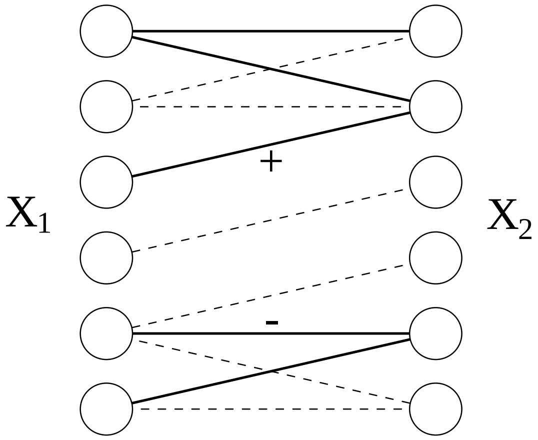


Figure 7.13: This figure is adapted from [Blum and Mitchell, 1998]. Left-hand nodes are points in X_1, right-hand nodes are points in X_2. Solid edges represent examples in the training or test sample. Training examples are labeled + or -, while test examples are unlabeled. Dashed edges represent examples with non-zero probability. However, they are not observed in the training or test sample.

The use of EM on labeled and unlabeled data was also explored in other domains and for other types of mixture models [Miller and Uyar, 1997][Shahshahani and Landgrebe, 1994]. For a restricted class of Gaussian mixtures Ratsaby and Venkatesh showed theoretically that unlabeled data reduces the required amount of labeled data [Ratsaby and Venkatesh, 1995].

7.6.2 Co-Training

Blum and Mitchell's work on co-training [Blum and Mitchell, 1998] uses unlabeled data in a particular setting. They exploit the fact that, for some problems, each example can be described by multiple representations. WWW-pages, for example, can be represented as the text on the page and/or the anchor texts on the hyperlinks pointing to this page. Blum and Mitchell develop a boosting scheme which exploits a conditional independence between these representations. A similar setting of using multiple representations is also explored in [de Sa, 1993].

The following argument shows that transductive learning to some extent subsumes co-training. Consider the rote co-training setting as defined in [Blum and Mitchell, 1998]. Each example $\vec{x} = (\vec{x}^{(1)}, \vec{x}^{(2)})$ is described by a pair of representations with points from X_1 and X_2. Assume that the number of points in X_1 and X_2 is N (i. e. $|X_1| = |X_2| = N$). X_1 and X_2 each have a matching concept class C_1 and C_2. In rote co-training learning is reduced to a look-up table, so that all concepts $C_1 = 2^{X_1}$ and $C_2 = 2^{X_2}$ can be represented. A target concept h is a pair $(h_1, h_2) \in C_1 \times C_2$. Blum and Mitchell require that there is no noise and that the target concept is fully compatible with the distribution of examples. This means that all examples with a probability of occurrence greater than zero are classified consistently and correctly by both parts of the target concept (i.e. $h_1(\vec{x}^{(1)}) = h_2(\vec{x}^{(2)})$).

Blum and Mitchell present and analyze an algorithm that can handle rote co-training effectively [Blum and Mitchell, 1998]. It is based on a representation of the training sample as a bipartite graph. An example is given in Figure 7.13. Each (labeled and unlabeled) example $\vec{x} = (\vec{x}^{(1)}, \vec{x}^{(2)})$ defines an edge in the graph. It is easy to see that examples in the same connected component of the graph must have the same label. So one labeled training example is sufficient for labeling the whole connected component.

The following shows that their algorithm behaves identically to a transductive support vector machine in this restricted setting. For any sample of labeled and unlabeled data, a TSVM outputs the same classification of the unlabeled data as the co-training algorithm of Blum and Mitchell. Introduce a new feature space X' with one feature for each point in X_1 and X_2. Each co-training example $\vec{x} = (\vec{x}^{(1)}, \vec{x}^{(2)})$ is mapped into this new features space by setting the feature values corresponding to $\vec{x}^{(1)}$ and $\vec{x}^{(2)}$ to 1, all other feature values to 0.[1] This means that every example $\vec{x}'$ has exactly two 1's in it. Note that a connected component now corresponds to a minimal set of vectors that are orthogonal to all other vectors not in that set.

A first hint that TSVMs apply to rote co-training is given by the following observation. In X' the co-training problem can be sufficiently modeled as a TCat-concept. Denote with $|f_1^+|$ and $|f_2^+|$ the number of points $\vec{x}^{(1)}$ and $\vec{x}^{(2)}$ that occur only in positive examples. Similarly, denote $|f_1^-|$ and $|f_2^-|$ for negative examples. Then

$$\begin{array}{lll} TCat(& [1:0:|f_1^+|], [1:0:|f_2^+|], & \#\ \text{positive examples} \\ & [0:1:|f_1^-|], [0:1:|f_2^-|] & \#\ \text{negative examples} \\ ) & & \end{array} \tag{7.12}$$

describes the co-training problem in X'. The following theorem elaborates on the connection between rote co-training concepts and their margin.

Theorem 13 (Margin of Rote Co-Training Concepts) *After the mapping, every fully compatible co-training problem in the rote learning setting is linearly separable through the origin with a margin ratio R^2/δ^2 of at most*

$$\frac{R^2}{\delta^2} \leq N \tag{7.13}$$

Proof *The squared length of each weight vector is 2, since each training example in X' contains exactly two elements equal to 1. Therefore $R^2 = 2$.*

The margin δ of a hyperplane passing through the origin is related to the solution of the following optimization problem.

$$\frac{1}{\delta^2} \quad = \quad \min \vec{w} \cdot \vec{w} \ \ s.\ t.\ \forall i \in [1..n] : y_i \vec{w} \cdot \vec{x}_i \geq 1 \tag{7.14}$$

The following construction for $\vec{w}$ ensures that all constraints are fulfilled. For all points $\vec{x}^{(1)}$ and $\vec{x}^{(2)}$) associated with positive examples, set the corresponding components of $\vec{w}$ to 0.5. Similarly set them to -0.5 for negative examples. Then for positive examples it holds that $\vec{w} \cdot \vec{x}_i = 1$. For negative examples $\vec{w} \cdot \vec{x}_i = -1$. Therefore the conditions in (7.14) are always fulfilled and the constructed $\vec{w}$ is a feasible point. The squared Euclidian norm of $\vec{w}$ is $2N(\frac{1}{2})^2$, which is an upper bound on the solution of (7.14). ∎

The following lemma is somewhat stronger. It establishes that the largest margin separation is always between two connected components. This implies that a TSVM never applies different labels to examples from the same connected component.

[1] Note that this does not incur computational problems for an SVM. While X' is very high dimensional, the feature vectors are sparse.

Lemma 9 (Maximum-Margin Labeling of Connected Components) *Given is a set of vectors $\vec{x}_1, ..., \vec{x}_n$. Each $\vec{x}$ has exactly two features equal to one, all others are zero. If none of these vectors is orthogonal to all others (i.e. $\vec{x}_1, ..., \vec{x}_n$ form a connected component in the rote co-training problem), then the unbiased hyperplane that has the maximum margin always assigns the same label to all such examples.*

Proof *It is sufficient to show that the objective function of*

$$W(\vec{\alpha}) = max_{0 \leq \vec{\alpha}} 1^T \vec{\alpha} - \frac{1}{2} \vec{\alpha}^T diag(\vec{y}) Q diag(\vec{y}) \vec{\alpha} \tag{7.15}$$

at the solution is smallest for a uniform labeling of the examples $\vec{x}_1, ..., \vec{x}_n$ (i.e. $\vec{y}^{+T} = (+1, ..., +1)^T$ or $\vec{y}^{-T} = (-1, ..., -1)^T$). Let $\vec{\alpha}^+$ be the solution for the uniform labeling $\vec{y}^+$. Note that both uniform labelings have the same solution since $diag(\vec{y}^+) Q diag(\vec{y}^+) = diag(\vec{y}^-) Q diag(\vec{y}^-)$. So the same argument given below for $\vec{y}^+$ also holds for $\vec{y}^-$. Now choose a different labeling $\vec{y}'$ by flipping $1 \leq f \leq n-1$ labels in $\vec{y}^+$. Denote the set of indizes of flipped examples by F, all others by N.

Let us first show that no such labeling $\vec{y}'$ can lead to a smaller objective than at the solution $\vec{\alpha}^+$ for $\vec{y}^+$. Since all elements in the Hessian Q are positive and $\vec{\alpha} \geq 0$, all terms in $\vec{\alpha}^T diag(\vec{y}^+) Q diag(\vec{y}^+) \vec{\alpha} = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i^+ y_j^+ \vec{x}_i \cdot \vec{x}_j$ are positive for a uniform labeling. Having a non-uniform labeling will flip the sign for some of these terms without changing their absolute value. Then $\vec{\alpha}^+$ is a feasible point for the labeling $\vec{y}'$ that has at least the same value of the objective. In the worst case the negative signs will occur exactly for those elements of the Hessian which are zero.

It remains to show that there is a strict increase in the objective for non-uniform labelings. Let SV^+ be the indices of the support vectors at the solution $\vec{\alpha}^+$. The following three cases can occur depending on how the labels $\vec{y}'$ are flipped.

Case $\vec{x}_i \cdot \vec{x}_j > 0$ for at least one $i \in SV^+ \cap F$ and $j \in SV^+ \cap N$, $SV^+ \cap F \neq \emptyset$ and $SV^+ \cap N \neq \emptyset$: *This means there is at least one $\alpha_i \alpha_j y_i^+ y_j^+ \vec{x}_i \cdot \vec{x}_j > 0$ leading to a strict decrease if one of the labels is flipped.*

Case $\vec{x}_i \cdot \vec{x}_j = 0$ for all $i \in SV^+ \cap F$ and $j \in SV^+ \cap N$, $SV^+ \cap F \neq \emptyset$ and $SV^+ \cap N \neq \emptyset$: *Then there exists a non-SV $\vec{x}_k$ that is not orthogonal to at least one $\vec{x}_i, i \in SV^+ \cap F$. Without loss of generality, let us assume $k \in N$. Since $\vec{x}_k$ contains only two non-zero features, it must share one with a SV in F and one with a SV in N. This follows from the observation that both $\sum_{i \in F} \alpha_i y_i^+ \vec{x}_i \cdot \vec{x}_k < 1$ and $\sum_{i \in N} \alpha_i y_i^+ \vec{x}_i \cdot \vec{x}_k < 1$. If one of them was larger or equal to one, then there was a SV j with $\sum_{i \in SV} \alpha_i y_i^+ \vec{x}_i \cdot \vec{x}_j > 1$, which is a contradiction. Using $\vec{\alpha}^+$ as a feasible point for the new labeling $\vec{y}'$, one can find the optimal value of α'_k while keeping all other $\vec{\alpha}^+$ konstant. Finding this α'_k by differentiation leads to*

$$\alpha'_k = \frac{1 - \sum_{i=1}^n \alpha_i^+ y'_i y'_k \vec{x}_i \cdot \vec{x}_k}{\vec{x}_k \cdot \vec{x}_k} \tag{7.16}$$

α'_k is always positive, since $\sum_{i=1}^n \alpha_i^+ y'_i y'_k \vec{x}_i \cdot \vec{x}_k < 1$. This inequality holds since there is no (positive) contribution from the set F any more. This means α'_k is a feasible point. Substituing α'_k into the optimization problem and substracting from the objective of the uniform labeling solution, it follows that at the new feasible point the objective is

$$\frac{1}{2} \frac{(1 - \sum_{i=1}^n \alpha_i^+ y'_i y'_k \vec{x}_i \cdot \vec{x}_k)^2}{\vec{x}_k \cdot \vec{x}_k} \tag{7.17}$$

larger than at the uniform labeling solution.

Case $SV^+ \cap F = \emptyset$ or $SV^+ \cap N = \emptyset$: *This means that there is at least one non-SV k that has a flipped label, while all SV are uniformly labeled. Again, we calculate the optimal α'_k while keeping all other α_i fixed using (7.16). α'_k is always positive and therefore feasible, since $\sum_{i=1}^n \alpha_i^+ y'_i y'_k \vec{x}_i \cdot \vec{x}_k < -1$. As for the previous case, this α'_k leads to an increase in the objective function.* ▮

Using this lemma it is easy to show that the transductive SVM labels the unlabeled examples exactly like the co-training algorithm of Blum and Mitchell.

Theorem 14 (Transductive SVMs for Rote Co-Training) *For rote co-training problems, the transductive SVM labels the test examples like the co-training algorithm of Blum and Mitchell.*

Proof *The transductive SVM labels all test examples correctly, for which there is at least one labeled training example in the same connected component. To show this, one can decompose the optimization problem into the sum of independent optimization problems. There is one such subproblem for each connected component $a, b, c, ...$.*

$$\begin{aligned}
W(\vec{\alpha}) &= max_{0 \le \vec{\alpha}, \vec{y} \in \{-1,+1\}} 1^T \vec{\alpha} - \frac{1}{2} \vec{\alpha}^T diag(\vec{y}) Q diag(\vec{y}) \vec{\alpha} && (7.18)\\
&= max_{0 \le \vec{\alpha}_a, \vec{y}_a \in \{-1,+1\}} 1^T \vec{\alpha}_a - \frac{1}{2} \vec{\alpha}_a^T diag(\vec{y}_a) Q_{aa} diag(\vec{y}_a) \vec{\alpha}_a && (7.19)\\
&\quad + max_{0 \le \vec{\alpha}_b, \vec{y}_b \in \{-1,+1\}} 1^T \vec{\alpha}_b - \frac{1}{2} \vec{\alpha}_b^T diag(\vec{y}_b) Q_{bb} diag(\vec{y}_b) \vec{\alpha}_b\\
&\quad + max_{0 \le \vec{\alpha}_c, \vec{y}_c \in \{-1,+1\}} 1^T \vec{\alpha}_c - \frac{1}{2} \vec{\alpha}_c^T diag(\vec{y}_c) Q_{cc} diag(\vec{y}_c) \vec{\alpha}_c\\
&\quad \vdots
\end{aligned}$$

The subproblems are independent, since the vectors of different connected components are orthogonal. Lemma 9 establishes that each such subproblem has a solution with uniform labeling. Since the transductive SVM fits the training data perfectly, all examples in a connected component are classified correctly. On all other test examples, the transductive SVM assigns an unspecified label. This is equivalent to the rote co-training algorithm in [Blum and Mitchell, 1998]. ▮

The theorem shows that transductive SVMs can be suitable for co-training, since the rote co-training setting formulates a sufficient condition for large-margin separation. While it describes only the rote learning case, a straightforward extension of the TSVM to general co-training is the following. Simply concatenate both feature spaces X_1 and X_2 into one large feature space X'. Then use a TSVM on X'. It is easy to see that linear separability in X_1 and X_2 also leads to linear separability in X' given full compatibility. Nevertheless, it is an open question whether the TSVM loses information through the concatenation into one feature space that could be exploited by a "native" co-training algorithm.

A clear advantage of co-training is that it is not restricted to a particular base learning algorithm. So it is possible to plug in the learner most appropriate for each task. However, TSVMs also provide some modeling freedom through specially designed kernels.

7.6.3 Other Work on Transduction

Early empirical results using transduction can be found in [Vapnik and Sterin, 1977]. They also introduces the general idea of transduction and its application to support vector machines. More recently, Bennett [Bennett, 1999] and Fung & Mangasarian [Fung and Mangasarian, 1999] presented results for a modified TSVM formulation. For ease of computation, they conducted the experiments only for a linear-programming approach which minimizes the L_1 norm instead of L_2 and prohibits the use of kernels. This approach showed small improvements for some of the standard UCI data sets.

Wu et al. applied the idea of transduction to decision trees [Wu et al., 1999]. They use the conventional top-down approach for growing decision trees. The algorithm differs in its splitting criterion. In each node Wu et al. use a perceptron that maximizes margin not only for the training but also for the test data. Again, they find small improvements for some of the standard UCI data sets.

Connecting to concepts of algorithmic randomness, [Gammerman et al., 1998][Vovk et al., 1999][Saunders et al., 1999] presented an approach to estimating the confidence of a prediction based on a transductive setting. A similar goal using a Bayesian approach is pursued in [Graepel et al., 2000]. Since their primary aim is not a reduced error rate in general, but a measure of confidence for a particular prediction, they consider only test sets with exactly one example.

The aspect of transduction can also be found in [Cataltepe and Magdon-Ismail, 1998]. They augment training error in linear regression with two additional terms. The first one depends on the location of the training examples, while the second one is its analog for the test examples. Both terms substract to zero when the hypothesis behaves similarly on both the training and the test data. Otherwise, they penalize that hyperplane. In its general spirit, this is similar to the approach presented in Section 7.5.

7.7 Summary and Conclusions

This chapter has introduced a transductive approach to text classification. This framework is fundamentally different from the conventional inductive approach to learning text classifiers. While an inductive learner aims to find a classification rule that has a low error on the whole distribution of examples, a transductive learner minimizes prediction error for a given set of test examples. For tasks like relevance feedback, where the test set is known, a transductive framework is preferable over an inductive approach, since it models the learning problem more closely.

Transductive support vector machines are the natural extension of SVMs to the transductive setting. The TSVM maximizes margin not only on the training, but also on the test set. Exploiting the particular statistical properties of text, this chapter has identified the margin of separating hyperplanes as a natural way to encode prior knowledge for learning text classifiers. By taking a transductive instead of an inductive approach, the test set can be used as an additional source of information about margins. Connecting to the theoretical model of text classification from Chapter 4, the analysis explains for the first time why and how the TSVM can take advantage of the margin on the test set.

This work presents empirical results on three test collections. On all data sets the transductive approach showed improvements over the currently best performing method, most substantially for small training samples and large test sets.

This chapter has also introduced a new method for verifying the labeling of the test data produced by the TSVM. It is based on a statistical hypothesis test against a necessary condition that the TSVM solution has to fulfill for the correct labeling of the test examples. The criterion can be computed at essentially no extra cost and without additional data. Experiments show that it can effectively verify the validity of the TSVM predictions.

There are still many open questions regarding transductive inference and SVMs. Particularly interesting is a PAC-style model for transductive inference. How does the sample complexity behave for both the training and the test set? What is the relationship between the concept and the instance distribution? What other concept classes benefit from transductive learning as well? Regarding text classification in particular, is there a better basic representation for text, aligning margin and learning bias even better? Finally, the transductive classification implicitly defines a classification rule. Is it possible to use this classification rule in an inductive fashion and will it also perform well on new test examples?

Part III

Algorithms

Chapter 8

Training Inductive Support Vector Machines

Training a support vector machine (SVM) leads to a quadratic optimization problem with bound constraints and one linear equality constraint. Despite the fact that this type of problem is well understood in principle, there are many issues to be considered in designing an SVM learner. In particular, for large learning tasks with many training examples, off-the-shelf optimization techniques for general quadratic programs such as Newton, Quasi Newton, etc., quickly become intractable in their memory and time requirements.

This chapter derives a training algorithm for support vector machines that addresses the problem of large tasks. It is based on a decomposition that reduces the optimization problem into a series of smaller tasks. Each small problem can be solved efficiently. A strategy for finding a good decomposition and a method for reducing the size of the problem by excluding irrelevant variables are the key elements of the algorithm. The algorithm is evaluated on several benchmark data sets. It is shown to be orders of magnitude faster than the conventional "chunking" algorithm [Boser et al., 1992]. While it can be used for any type of data, special emphasis is put on the discussion of learning text classifiers. In particular, the algorithm is designed so that its time complexity does not necessarily depend on of the dimensionality of the input space for sparse feature vectors.

The training algorithm is implemented in the software SVM^{light} (Version 3.10). Beginning in 1997, SVM^{light} has been available on the World Wide Web[1]. Since then it has had more than 350 registered users. It has been used in commercial applications and studies ranging from text classification [Yang and Liu, 1999, Neumann and Schmeier, 1999, Busemann et al., 2000, Kindermann et al., 2000], to image recognition [Lerner and Lawrence, 2000], bioinformatics [Moler et al., 2000], and intensive care monitoring [Morik et al., 2000], to studies against benchmarks [Syed et al., 1999].

8.1 Problem and Approach

Vapnik (e.g. [Vapnik, 1998]) and Chapter 3 show that training a support vector machine for the pattern recognition problem leads to the following quadratic optimization problem (QP).

[1] SVM^{light} is available at `http://www-ai.cs.uni-dortmund.de/svm_light`

Optimization Problem 11 (Soft-Margin SVM (dual))

$$\textit{minimize:} \qquad W(\vec{\alpha}) = -\sum_{i=1}^{n} \alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \tag{8.1}$$

$$\textit{subject to:} \qquad \sum_{i=1}^{n} y_i \alpha_i = 0 \tag{8.2}$$

$$\forall i : 0 \leq \alpha_i \leq C \tag{8.3}$$

The number of training examples is denoted by n. $\vec{\alpha}$ is a vector of n variables, where each component α_i corresponds to a training example $(\vec{x}_i, y_i)$. The solution of Optimization Problem 11 (OP11) is the vector $\vec{\alpha}^*$ for which (8.1) is minimized and the constraints (8.2) and (8.3) are fulfilled. Defining the matrix Q as $(Q)_{ij} = y_i y_j \mathcal{K}(\vec{x}_i, \vec{x}_j)$, this can equivalently be written as

$$\text{minimize:} \qquad W(\vec{\alpha}) = -\vec{\alpha}^T \vec{1} + \frac{1}{2}\vec{\alpha}^T Q \vec{\alpha} \tag{8.4}$$

$$\text{subject to:} \qquad \vec{\alpha}^T \vec{y} = 0 \tag{8.5}$$

$$\vec{0} \leq \vec{\alpha} \leq C\vec{1} \tag{8.6}$$

The size of the optimization problem depends on the number of training examples n. Since the size of the matrix Q is n^2, for learning tasks with 10000 training examples and more it becomes impossible to keep Q in memory. Many standard implementations of QP solvers require explicit storage of Q which prohibits their application. An alternative would be to recompute Q every time it is needed. But this becomes prohibitively expensive if Q is needed often. The following will show that even computing Q completely only once should be avoided.

One approach to making the training of SVMs on problems with many training examples tractable is to decompose the problem into a series of smaller tasks. SVM^{light} uses the decomposition idea of [Osuna et al., 1997a]. This decomposition splits Optimization Problem 11 into an inactive and an active part - the so called “working set”. The main advantage of this decomposition is that it suggests algorithms with memory requirements linear in the number of training examples and linear in the number of SVs. One potential disadvantage is that these algorithms may need a long training time. To tackle this problem, this chapter proposes an algorithm which incorporates the following ideas:

- An efficient and effective method for selecting the working set.
- Successive “shrinking” of the optimization problem. This exploits the property that many SVM learning problems have
  - many fewer support vectors (SVs) than training examples.
  - many SVs which have an α_i at the upper bound C.
- Computational improvements like caching and incremental updates of the gradient and the termination criteria.

This chapter is structured as follows. First, a generalized version of the decompositon algorithm of [Osuna et al., 1997a] is introduced. This identifies the problem of selecting the working set, which is addressed in the following section. In Section 8.4 a method for “shrinking” OP11 is presented and Section 8.5 describes the computational and implementational approach of SVM^{light} . Finally, experimental results on text-classification task and two benchmark data sets are discussed to evaluate the approach.

8.2 General Decomposition Algorithm

This section presents a generalized version of the decomposition strategy proposed by [Osuna et al., 1997a]. This strategy uses a decomposition similar to those used in *active set* strategies (see [Gill et al., 1981]) for the case that all inequality constraints are simple bounds. In each iteration the variables α_i of OP11 are split into two categories:

- the set B of free variables
- the set N of fixed variables

Free variables are those which can be updated in the current iteration, whereas fixed variables are temporarily fixed at a particular value. The set of free variables will also be referred to as the working set. The working set has a constant size q much smaller than n.

The algorithm works as follows:

```
• While the optimality conditions are violated
    – Select $q$ variables for the working set $B$.  The remaining
      $n-q$ variables are fixed at their current value.
    – Decompose problem and solve QP-subproblem:  optimize $W(\vec{\alpha})$ on $B$.
• Terminate and return $\vec{\alpha}$.
```

How can the algorithm detect that it has found the optimal value for $\vec{\alpha}$? Since OP11 is guaranteed to have a positive semi-definite Hessian Q and all constraints are linear, OP11 is a convex optimization problem. For this class of problems the following Kuhn-Tucker conditions are necessary and sufficient conditions for optimality. Denoting the Lagrange multiplier for the equality constraint 8.5 with λ^{eq} and the Lagrange multipliers for the lower and upper bounds 8.6 with $\vec{\lambda}^{lo}$ and $\vec{\lambda}^{up}$, $\vec{\alpha}$ is optimal for OP11, if there exist λ^{eq}, $\vec{\lambda}^{lo}$, and $\vec{\lambda}^{up}$, so that (Karush-Kuhn-Tucker Conditions , see [Werner, 1984]):

$$\begin{array}{rcll}
 & g(\vec{\alpha}) + (\lambda^{eq}\vec{y} - \vec{\lambda}^{lo} + \vec{\lambda}^{up}) & = \vec{0} & (8.7) \\
\forall i \in [1..n]: & \lambda_i^{lo}(-\alpha_i) & = 0 & (8.8) \\
\forall i \in [1..n]: & \lambda_i^{up}(\alpha_i - C) & = 0 & (8.9) \\
 & \vec{\lambda}^{lo} & \geq \vec{0} & (8.10) \\
 & \vec{\lambda}^{up} & \geq \vec{0} & (8.11) \\
 & \vec{\alpha}^T\vec{y} & = 0 & (8.12) \\
\vec{0} \leq & \vec{\alpha} & \leq C\vec{1} & (8.13)
\end{array}$$

$g(\vec{\alpha})$ is the vector of partial derivatives at $\vec{\alpha}$. For OP11 this is

$$g(\vec{\alpha}) = -\vec{1} + Q\vec{\alpha} \qquad (8.14)$$

If the optimality conditions do not hold, the algorithm decomposes Optimization Problem 11 and solves the smaller QP-problem arising from this. The decomposition assures that this will lead to progress in the objective function $W(\vec{\alpha})$, if the working set B fulfills some minimum requirements (see [Osuna et al., 1997b]). In particular, OP11 is decomposed by separating the variables in the working set B from those which are fixed (N). Let us assume $\vec{\alpha}$, $\vec{y}$, and Q are properly arranged with respect to B and N, so that

$$\vec{\alpha} = \left| \begin{array}{c} \vec{\alpha}_B \\ \vec{\alpha}_N \end{array} \right| \qquad \vec{y} = \left| \begin{array}{c} \vec{y}_B \\ \vec{y}_N \end{array} \right| \qquad Q = \left| \begin{array}{cc} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{array} \right| \tag{8.15}$$

Since Q is symmetric (in particular $Q_{BN} = Q_{NB}^T$), we can write the following:

Optimization Problem 12 (Soft-Margin SVM on Working Set B)

$$\begin{aligned} \textit{minimize:} \quad & W(\vec{\alpha}_B) = -\vec{\alpha}_B^T(\vec{1} - Q_{BN}\vec{\alpha}_N) + \frac{1}{2}\vec{\alpha}_B^T Q_{BB}\vec{\alpha}_B + \\ & \qquad \frac{1}{2}\vec{\alpha}_N^T Q_{NN}\vec{\alpha}_N - \vec{\alpha}_N^T\vec{1} && (8.16) \\ \textit{subject to:} \quad & \vec{\alpha}_B^T\vec{y}_B + \vec{\alpha}_N^T\vec{y}_N = 0 && (8.17) \\ & \vec{0} \le \vec{\alpha} \le C\vec{1} && (8.18) \end{aligned}$$

Since the variables in N are fixed, the terms $\frac{1}{2}\vec{\alpha}_N^T Q_{NN}\vec{\alpha}_N$ and $-\vec{\alpha}_N^T\vec{1}$ are constant. They can be omitted without changing the solution of OP12. OP12 is a positive semi-definite quadratic programming problem which is small enough be solved by most off-the-shelf methods. It is easy to see that changing the α_i in the working set to the solution of OP12 is the optimal step on B. So fast progress depends heavily on whether the algorithm can select good working sets.

8.3 Selecting a Good Working Set

When selecting the working set, it is desirable to select a set of variables such that the current iteration will make much progress towards the minimum of $W(\vec{\alpha})$. The following proposes a strategy based on Zoutendijk's method (see [Zoutendijk, 1970]), which uses a first-order approximation to the target function. The idea is to find a steepest feasible direction $\vec{d}$ of descent which has only q non-zero elements. The variables corresponding to these elements will compose the current working set.

This approach leads to the following optimization problem. Its solution describes, which variables enter the working set at each iteration.

Optimization Problem 13 (Working Set Selection)

$$\begin{aligned} \textit{minimize:} \quad & V(\vec{d}) = g(\vec{\alpha}^{(t)})^T\vec{d} && (8.19) \\ \textit{subject to:} \quad & \vec{y}^T\vec{d} = 0 && (8.20) \\ & d_i \ge 0 \qquad \textit{for i: } \alpha_i = 0 && (8.21) \\ & d_i \le 0 \qquad \textit{for i: } \alpha_i = C && (8.22) \\ & -\vec{1} \le \vec{d} \le \vec{1} && (8.23) \\ & |\{d_i : d_i \ne 0\}| = q && (8.24) \end{aligned}$$

The objective (8.19) states that a direction of descent is wanted. A direction of descent has a negative dot-product with the vector of partial derivatives $g(\vec{\alpha}^{(t)})$ at the current point $\vec{\alpha}^{(t)}$. Constraints (8.20), (8.21), and (8.22) ensure that the direction of descent is projected along the equality constraint (8.5) and obeys the active bound constraints. Constraint (8.23) normalizes the descent vector to make the optimization problem well-posed. Finally, the last constraint (8.24) states that the direction of descent shall only involve q variables. The variables with non-zero d_i are included into the working set B. This way we select the working set with the steepest feasible direction of descent.

8.3.1 Convergence

The selection strategy, the optimality conditions, and the decomposition together specify the optimization algorithm. A minimum requirement this algorithm has to fulfill is that it

- terminates only when the optimal solution is found
- if not at the solution, takes a step towards the optimum

The first requirement can easily be fulfilled by checking the (necessary and sufficient) optimality conditions (8.7) to (8.13) in each iteration. For the second one, let us assume the current $\vec{\alpha}^{(t)}$ is not optimal. Then the selection strategy for the working set returns an optimization problem of type OP12. Since by construction for this optimization problem there exists a $\vec{d}$ which is a feasible direction for descent, we know using the results of [Zoutendijk, 1970] that the current OP12 is non-optimal. So optimizing OP12 will lead to a lower value of the objective function of OP12. Since the solution of OP12 is also feasible for OP11 and due to the decomposition (8.16), we also get a lower value for OP11. This means we get a strict descent in the objective function in each iteration.

However, as elaborated in [Chang et al., 1999], even with a strict descent convergence is not guaranteed. If the step towards the optimum can be arbitrarily small, the algorithm does not necessarily reach the solution. In particular, using $\vec{d}$ as the search direction and computing the maximum step size like prescribed by Zoutendijk does not necessarily lead to a convergent algorithm on all problems [Wolfe, 1972]. However, in the previous section $\vec{d}$ is not directly used as the search direction, but merely for selecting the working set. On the resulting working set the algorithm makes an optimal step by solving the decomposed quadratic program. Recently, Chang et al. analyzed the working-set-selection strategy proposed above [Chang et al., 1999]. They present a proof that a variant of the working-set-selection strategy does always converge. A different convergence proof that directly applies to the selection criterion of Optimization Problem 13 is given in [Lin, 2000].

8.3.2 How to Compute the Working Set

The solution to OP13 is easy to compute using a simple strategy. Let $\omega_i = y_i g_i(\vec{\alpha}^{(t)})$ and sort all α_i according to ω_i in decreasing order. Let us furthermore require that q is an even number. Successively pick the $q/2$ elements from the top of the list for which $0 < \alpha_i^{(t)} < C$, or $d_i = -y_i$ obeys (8.21) and (8.22). Similarly, pick the $q/2$ elements from the bottom of the list for which $0 < \alpha_i^{(t)} < C$, or $d_i = y_i$ obeys (8.21) and (8.22). These q variables compose the working set.

8.4 Shrinking: Reducing the Number of Variables

For many tasks the number of SVs is much smaller than the number of training examples. If it were known a priori which of the training examples turn out to be SVs, it would be sufficient to train just on those examples and still get the same result. This would make OP11 smaller and faster to solve, since we could save time and space by not needing parts of the Hessian Q which do not correspond to SVs.

Similarly, for noisy problems there are often many SVs with an α_i at the upper bound C. Let us call these support vectors "bounded support vectors" (BSVs). Similar arguments as for the non-support vectors apply to BSVs. If it was known a priori which of the training examples turn out as BSVs, the corresponding α_i could be fixed at C, leading to a new optimization problem with fewer variables.

During the optimization process it often becomes clear fairly early that certain examples are unlikely to end up as SVs or that they will be BSVs. By eliminating these variables from OP11, we get a smaller problem OP11$'$ of size n'. From OP11$'$ we can construct the solution of OP11. Let X denote those indices corresponding to unbounded support vectors, Y those indexes which correspond to BSVs, and Z the indices of non-support vectors. The transformation from OP11 to OP11$'$ can be done using a decomposition similar to (8.16). Let us assume $\vec{\alpha}$, $\vec{y}$, and Q are properly arranged with respect to X, Y, and Z, so that we can write

$$\vec{\alpha} = \begin{vmatrix} \vec{\alpha}_X \\ \vec{\alpha}_Y \\ \vec{\alpha}_Z \end{vmatrix} = \begin{vmatrix} \vec{\alpha}_X \\ C\vec{1} \\ \vec{0} \end{vmatrix} \qquad \vec{y} = \begin{vmatrix} \vec{y}_X \\ \vec{y}_Y \\ \vec{y}_Z \end{vmatrix} \qquad Q = \begin{vmatrix} Q_{XX} & Q_{XY} & Q_{XZ} \\ Q_{YX} & Q_{YY} & Q_{YZ} \\ Q_{ZX} & Q_{ZY} & Q_{ZZ} \end{vmatrix} \tag{8.25}$$

The decomposition of $W(\vec{\alpha})$ is

$$\text{minimize:} \quad W(\vec{\alpha}_X) = -\vec{\alpha}_X^T(\vec{1} - (Q_{XY}\vec{1}) \cdot C) + \frac{1}{2}\vec{\alpha}_X^T Q_{XX}\vec{\alpha}_X + \frac{1}{2}C\vec{1}^T Q_{YY} C\vec{1} - |Y|C \tag{8.26}$$

$$\text{subject to:} \quad \vec{\alpha}_X^T \vec{y}_X + C\vec{1}^T \vec{y}_Y = 0 \tag{8.27}$$

$$\vec{0} \leq \vec{\alpha_X} \leq C\vec{1} \tag{8.28}$$

Since $\frac{1}{2}C\vec{1}^T Q_{YY} C\vec{1} - |Y|C$ is constant, it can be dropped without changing the solution. So far it is not clear how the algorithm can identify which examples can be eliminated. It is desirable to find conditions which indicate early in the optimization process that certain variables will end up at a bound. Since sufficient conditions are not known, a heuristic approach based on Lagrange multiplier estimates is used.

At the solution, the Lagrange multiplier of a bound constraint indicates how much the variable "pushes" against that constraint. A strictly positive value of a Lagrange multiplier of a bound constraint indicates that the variable is optimal at that bound. At non-optimal points, an estimate of the Lagrange multiplier can be used. Let A be the current set of α_i fulfilling $0 < \alpha_i < C$. By solving (8.7) for λ^{eq} and averaging over all α_i in A, we get the estimate (8.29) for λ^{eq}.

$$\lambda^{eq} = \frac{1}{|A|} \sum_{i \in A} \left[y_i - \sum_{j=1}^{n} \alpha_j y_j \mathcal{K}(\vec{x_i}, \vec{x_j}) \right] \tag{8.29}$$

Note the equivalence of λ^{eq} and b in the classification rule (3.21). Since variables α_i cannot be both at the upper and the lower bound simultanously, the multipliers of the bound constraints can now be estimated by

$$\lambda_i^{lo} = y_i \left(\left[\sum_{j=1}^{n} \alpha_j y_j \mathcal{K}(\vec{x_i}, \vec{x_j}) \right] + \lambda^{eq} \right) - 1 \tag{8.30}$$

for the lower bounds and by

$$\lambda_i^{up} = -y_i \left(\left[\sum_{j=1}^{n} \alpha_j y_j \mathcal{K}(\vec{x_i}, \vec{x_j}) \right] + \lambda^{eq} \right) + 1 \tag{8.31}$$

for the upper bounds. Let us consider the history of the Lagrange multiplier estimates over the last h iterations. If the estimate (8.30) or (8.31) was positive (or above some positive threshold Θ) at each of the last h iterations, it is likely that this will be true at the optimal solution, too. These variables are eliminated using the decomposition from above. This means that these variables are fixed and neither the gradient nor the optimality conditions are computed. This leads to a substantial reduction in the number of kernel evaluations.

Since this heuristic can fail, the optimality conditions for the excluded variables are checked after convergence of OP11′. As described in the following section this can be done more efficiently for linear SVMs than for non-linear kernels. Therefore, the values for h and Θ should be chosen more conservatively for non-linear SVMs. If the optimality conditions for some variables are not fulfilled, the full problem is reoptimized starting from the solution of OP11′.

8.5 Efficient Implementation

While the previous sections dealt with algorithmic issues, there are still a lot of open questions to be answered before having an efficient implementation. This section addresses these implementational issues.

8.5.1 Termination Criteria

There are two obvious ways to define termination criteria which fit nicely into the algorithmic framework presented above. First, the solution of OP13 can be used to define a necessary and sufficient condition for optimality. If (8.19) equals 0, OP11 is solved with the current $\vec{\alpha}^{(t)}$ as solution.

However, for this stopping criterion it is difficult to specify the desired precision in a meaningful and intuitive way. Therefore, SVM^{light} goes another way and uses a termination criterion derived from the optimality conditions (8.7)-(8.13). Using the same reasoning as for (8.29)-(8.31), the following conditions with $\epsilon = 0$ are equivalent to (8.7)-(8.13).

$$\forall i \text{ with } 0 < \alpha_i < C: \quad \lambda^{eq} - \epsilon \leq y_i - [\textstyle\sum_{j=1}^{n} \alpha_j y_j \mathcal{K}(\vec{x}_i, \vec{x}_j)] \leq \lambda^{eq} + \epsilon \tag{8.32}$$

$$\forall i \text{ with } \alpha_i = 0: \quad y_i([\textstyle\sum_{j=1}^{n} \alpha_j y_j \mathcal{K}(\vec{x}_i, \vec{x}_j)] + \lambda^{eq}) \geq 1 - \epsilon \tag{8.33}$$

$$\forall i \text{ with } \alpha_i = C: \quad y_i([\textstyle\sum_{j=1}^{n} \alpha_j y_j \mathcal{K}(\vec{x}_i, \vec{x}_j)] + \lambda^{eq}) \leq 1 + \epsilon \tag{8.34}$$

$$\vec{\alpha}^T \vec{y} = 0 \tag{8.35}$$

The optimality conditions (8.32), (8.33), and (8.34) are very natural since they reflect the constraints of the primal optimization problem (3). In practice these conditions need not be fulfilled with high accuracy. Using a tolerance of $\epsilon = 0.001$ is acceptable for most tasks. Using a higher accuracy did not show improved generalization performance on the tasks tried, but lead to considerably longer training time.

8.5.2 Computing the Gradient and the Termination Criteria Efficiently

The efficiency of the optimization algorithm greatly depends on how efficiently the "housekeeping" in each iteration can be done. The following quantities are needed in each iteration.

- The vector of partial derivatives $g(\vec{\alpha}^{(t)})$ for selecting the working set.
- The values of the expressions (8.32), (8.33), and (8.34) for the termination criterion.

- The matrices Q_{BB} and Q_{BN} for the QP subproblem.

Fortunately, due to the decompositon approach, all these quantities can be computed or updated knowing only q rows of the Hessian Q. These q rows correspond to the variables in the current working set. The values in these rows are computed directly after the working set is selected and they are stored throughout the iteration. It is useful to introduce $\vec{s}^{(t)}$

$$s_i^{(t)} = \sum_{j=1}^{n} \alpha_j y_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \tag{8.36}$$

Knowing $\vec{s}^{(t)}$, the gradient (8.14) as well as in the termination criteria (8.32)-(8.34) can be computed very efficiently. When $\vec{\alpha}^{(t-1)}$ changes to $\vec{\alpha}^{(t)}$ the vector $\vec{s}^{(t)}$ needs to be updated. This can be done efficiently and with sufficient accuracy as follows:

$$s_i^{(t)} = s_i^{(t-1)} + \sum_{j \in B} (\alpha_j^{(t)} - \alpha_j^{(t-1)}) y_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \tag{8.37}$$

Note that only those rows of Q are needed which correspond to variables in the working set. The same is true for Q_{BB} and Q_{BN}, which are merely subsets of columns from these rows.

For the the linear kernel, the computation of $s_i^{(t)}$ can be sped up further using the following equality [Platt, 1999a]. It exploits the fact that in the linear case it is not necessary to compute the kernel for each support vector and each training example, but that the changes in the working set can be reduced to a single weight vector $\vec{w}_{\Delta}^{(t)}$.

$$\vec{w}_{\Delta}^{(t)} = \sum_{j \in B} (\alpha_j^{(t)} - \alpha_j^{(t-1)}) y_j \vec{x}_j \tag{8.38}$$

This weight vector is computed only once in each iteration. It reduces the update to

$$s_i^{(t)} = s_i^{(t-1)} + \vec{x}_i \cdot \vec{w}_{\Delta}^{(t)} \tag{8.39}$$

which can be much more efficiently done than computing the full expansion like in the non-linear case.

8.5.3 What are the Computational Resources Needed in each Iteration?

For non-linear kernels, most time in each iteration is spent on the kernel evaluations needed to compute the q rows of the Hessian. This step has a time complexity of $O(qlf)$, where f is the maximum number of non-zero features in any of the training examples. Using the stored rows of Q, updating $\vec{s}^{(t)}$ is done in time $O(ql)$. Setting up the QP subproblem requires $O(qq)$. Also the selection of the next working set, which includes computing the gradient, can be done in $O(ql)$. The highest memory requirements are due to storing the q rows of Q. Here $O(ql)$ floating point numbers need to be stored. Besides this, $O(q^2)$ is needed to store Q_{BB} and $O(l)$ to store $\vec{s}^{(t)}$, $\vec{\alpha}$, etc.

The linear case can be handled much more efficiently in two respects. Due to equation (8.38) it is not necessary to compute the q complete rows of the Hessian, but only the submatrix Q_{BB} that is the input to the core optimizer. This step has a time complexity of $O(q^2 f)$. Furthermore, updating $\vec{s}^{(t)}$ can be done in $O(qf)$ for computing the weight vector and $O(fl)$ for calculating the new $\vec{s}^{(t)}$. The memory requirements are smaller for the linear kernel as well, since it is not necessary to store the q rows of Q.

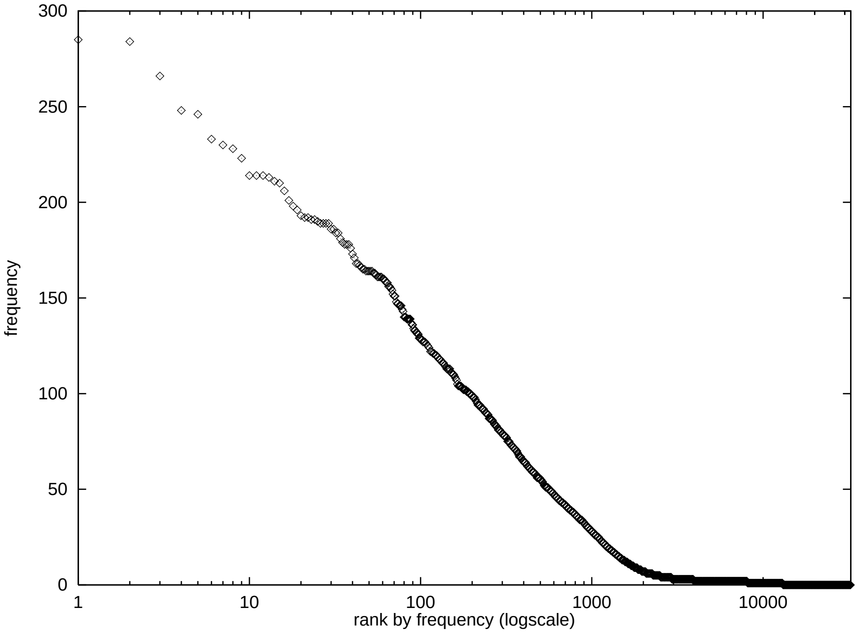


Figure 8.1: Frequency distribution of variables entering the working set for the income prediction task (32,562 examples, RBF-kernel with $\gamma = 0.05$ and $C = 1$). The working-set-size is $q = 10$.

Note that the time complexity of an iteration does not depend on the number of features. The non-linear kernels considered here all depend on a simple dot product between feature vectors. Using the sparse vector representation such dot products can be computed with a time complexity that depends only on the number of non-zero features. For text the number of non zero features is typically on the order of 10^2, while the total number of features is typically on the order of 10^4. In the linear case, a time complexity independent of the dimensionality of the feature space can be achieved using a single non-sparse vector for storing the weight vector $\vec{w}_{\Delta}^{(t)}$.

8.5.4 Caching Kernel Evaluations

As pointed out in the last section, for non-linear kernels the most expensive step in each iteration is the evaluation of the kernel to compute the q rows of the Hessian Q. Throughout the optimization process, eventual support vectors enter the working set multiple times. Figure 8.1 shows a typical example. For each example it plots the number of times it is selected into the working set. The examples are ranked by this frequency. The plot indicates that a small number of examples (approx. 2000) enters the working set very often, while the vast majority (approx. 30,000) is selected very infrequently or never. To avoid recomputation of the parts of the Hessian corresponding to frequently selected examples, SVM^{light} uses caching for non-linear kernels. This allows an elegant trade-off between memory consumption and training time.

SVM^{light} uses a least-recently-used caching strategy. When the cache is full, the element which has not been used for the greatest number of iterations, is removed to make room for the current row. Only those columns are computed and cached which correspond to active variables. After shrinking, the cache is reorganized accordingly.

8.5.5 How to Solve the QP on the Working Set

Currently, two core optimizers are available for solving the subproblems OP12. The default solver is based on the method of Hildreth and D'Espo [Hildreth, 1957][D'Espo, 1959] as described in [Wismer and Chattergy, 1978] (HIDEO). It is adapted to handle semi-definite problems by excluding variables that correspond to linearly dependent parts of the Hessian. The second solver is a primal-dual interior-point method (see [Vanderbei, 1994]) implemented by A. Smola [Smola, 1998] (LOQO). Other optimizers can easily be incorporated into SVM^{light} as well.

8.6 Related Work

The first approach to splitting large SVM learning problems into a series of smaller optimization tasks was proposed by [Boser et al., 1992]. It is known as the "chunking" algorithm (see also [Kaufman, 1999]). The algorithm starts with a random subset of the data, solves this problem, and iteratively adds examples which violate the optimality conditions. [Osuna et al., 1997b] prove formally that this strategy converges to the optimal solution. One disadvantage of this algorithm is that it is necessary to solve QP-problems scaling with the number of SVs. The decomposition of [Osuna et al., 1997a], which is used in the algorithm presented here, avoids this.

In parallel to the work presented here, an algorithm called Sequential Minimal Optimization (SMO) was explored for SVM training [Platt, 1998][Platt, 1999a]. It can be seen as a special case of the algorithm presented in this chapter, allowing only working sets of size 2. This restriction allows that the QP subproblems OP12 can be solved analytically, avoiding the need to implement a full QP solver. In addition, the algorithm differs in its working-set-selection-strategy. Instead of the steepest feasible descent approach presented here, SMO uses a set of heuristics motivated by the Karush-Kuhn-Tucker conditions. Nevertheless, these heuristics are likely to produce similar decisions in practice. While caching and shrinking were not part of the original SMO algorithm, they were recently transferred [Platt, 1999c]. Complementarily, handling linear SVMs in a special way was not part of the original SVM^{light} algorithm, but was later added as suggested in [Platt, 1999a].

Based on the original publication [Joachims, 1998a, Joachims, 1999b], the working-set-selection strategy from Section 8.3 was further developed and refined in several ways. Laskov extended the selection method to the SVM-regression problem [Laskov, 2000]. He shows that for regression it is also possible to achieve a similar speed-up as for the pattern recognition case considered here. Similarly, Chang and Lin [Chang and Lin, 1999] generalize the selection strategy to ν-SVMs [Schölkopf et al., 1999, Schölkopf et al., 2000]. A refined version of the working-set-selection strategy is proposed in [Hsu and Lin, 2000]. Their analysis reveals that convergence is slow when there are unnecessarily many free variables during the optimization process and when the optimizer zigzags during the final iterations. Zigzagging occurs when the working-set-selection strategy cycles between two or more working sets without making much progress. The extensions they propose tackle both problems and help reduce the number of iterations.

Other algorithms simplify the original SVM training problem to make it computationally easier to handle. In the successive overrelaxation approach [Mangasarian and Musicant, 1999] it is required that the hyperplane passes through the origin. The bias weight can only be mimicked by adding a new feature to the original feature space. This introduces an additional parameter that needs to be tuned and makes the SVM sensitive to translation and rotation of the data. However, the simpler formulation removes the equality constraint in the dual formulation so that it is possible to modify one variable at a time. This leads to a very simple update rule. A similar approach of modifying one variable at a time is followed with the Kernel-Adatron algorithm [Frieß et al., 1998].

Very recently, Mangasarian and Musicant explored a further simplified SVM formulation [Mangasarian and Musicant, 2000]. They require that the hyperplane not only passes through the origin, but also that training loss is measured squared (i.e. ξ^2) and not linear (i.e. ξ). Furthermore, they restrict themselves to linear support vector machines without kernels. Exploiting these simplifications, they present an algorithm that in each iteration requires matrix inversions scaling only with the number of features N, but not with the number of examples n. This results in an algorithm that is very fast for low-dimensional feature spaces. But since the number of features is very high in text classification, their algorithm is inappropriate for the problems considered in this dissertation.

While all algorithms discussed so far work directly with the primal/dual formulation of the SVM training problem, Keerthi et al. recently proposed a different approach [Keerthi et al., 1999]. They showed that the SVM training problem can be transformed into the problem of computing the nearest point between two convex polytopes. This transformation requires that the training data be linearly separable or, like above, that margin violations enter the objective function as ξ^2 instead of ξ. For nearest point problems, efficient algorithms already exist. Empirical results show that their training time is comparable to SMO.

8.7 Experiments

The following experiments evaluate the approach on the text-classification tasks Reuters, WebKB, and Ohsumed, as well as on two benchmark datasets. If not stated otherwise, the following experiments are done with SVM^{light} V3.10 using LOQO as the core optimizer. The cache size is 80 megabytes for non-linear kernels. The number of iterations h for the shrinking heuristic is 100 for non-linear kernels and 10 for the linear kernel, and OP11 is solved up to a precision of $\epsilon = 0.001$ in (8.32)-(8.34). The conventional chunking algorithm uses the projected conjugent gradient method as its QP solver [Platt, 1998].

8.7.1 Training Times for Reuters, WebKB, and Ohsumed

This section gives the training times for the text-classification experiments from Chapter 6. The experiments are conducted on a Pentium III/500Mhz with 128MB of a RAM running Linux. Tables 8.1, 8.2, and 8.3 show the results for the ten most frequent Reuters categories, the WebKB data set, and the five most frequent Ohsumed categories. Each table contains the training time and the number of support vectors for both the linear kernel and the RBF-kernel (3.34) with $\gamma = 0.1$. In addition to the total number of support vectors, the column BSV shows the number of support vectors at the upper bound C. The last line of each table gives the average training time over all categories. The working-set size is $q = 30$ for the linear SVM and $q = 10$ for the RBF-kernel.

The most notable observation is that a linear support vector machine can be trained much faster than an SVM with a non-linear kernel. While an average linear support vector machine can be trained within seconds for data sets with up to 10,000 examples and 35,000 features, the RBF-SVM requires minutes.

The results also show that there is a trend connected with the frequency of a category. Generally, the more positive examples there are for a category, the higher the training time is. The same is true for the number of support vectors. There are two other trends not observable from the tables. Training time tends to increase modestly for larger values

	linear, $C = 0.5$			RBF, $\gamma = 0.1$, $C = 1000$		
	Training Time	total SV	BSV	Training Time	total SV	BSV
earn	5.5	1826	653	196.1	1788	0
acq	6.6	2450	1124	241.2	2245	0
money-fx	4.9	1012	450	161.1	911	16
grain	4.4	1039	256	110.6	909	0
crude	2.7	846	322	110.8	838	4
trade	3.2	786	331	112.5	796	9
interest	4.5	763	369	138.9	624	22
ship	3.2	911	171	105.4	855	0
wheat	2.5	531	149	67.2	470	1
corn	2.7	573	151	67.4	533	0
average (all 90)	1.3			38.1		

Table 8.1: Training time (in cpu-seconds) for the ten most frequent Reuters categories (9603 examples) and average training time over all 90 categories.

	linear, $C = 0.5$			RBF, $\gamma = 0.1$, $C = 1000$		
	Training Time	total SV	BSV	Training Time	total SV	BSV
course	2.8	775	341	49.4	713	0
faculty	3.8	1351	837	84.0	1245	0
project	2.8	928	501	61.1	882	0
student	4.4	1389	866	81.2	1234	0
average	3.5			68.9		

Table 8.2: Training time (in cpu-seconds) for the WebKB categories (3957 examples) and average training time.

	linear, $C = 0.5$			RBF, $\gamma = 0.1$, $C = 1000$		
	Training Time	total SV	BSV	Training Time	total SV	BSV
Pathology	18.7	6052	3338	724.6	6267	0
Cardiovascular	11.8	3758	1628	393.5	3962	0
Neoplasms	11.3	3515	1140	363.2	3705	0
Nervous System	10.6	3439	1240	366.4	3760	0
Immunologic	9.0	2938	952	310.6	3098	0
average (all 23)	7.4			271.9		

Table 8.3: Training time (in cpu-seconds) for the five most frequent Ohsumed categories (10,000 examples) and average training time over all 23 categories.

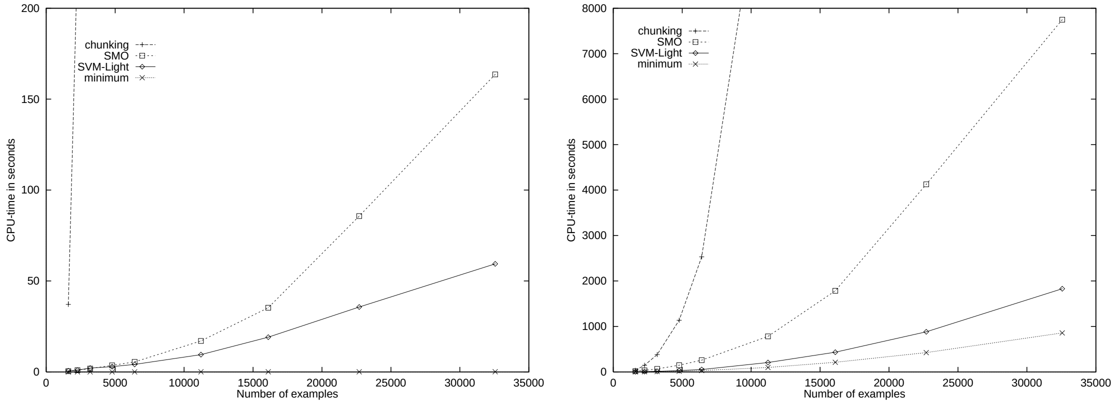


Figure 8.2: Training times for the linear SVM (left) and for the RBF-kernel (right) on the income prediction task. The times are repeated from tables 8.4 and 8.5.

of C independent of the choice of kernel. In addition, for non-linear kernels training time increases strongly with a larger γ for the RBF-kernel (3.34) and a higher degree of the polynomial kernel (3.33).

8.7.2 How does Training Time Scale with the Number of Training Examples?

For practical applications it is particularly important to know how training time scales with the training set size. This section first analyzes two benchmark tasks before considering an extended version of the Ohsumed data set. To be approximately comparable to previous studies [Platt, 1998] using a 266Mhz Pentium II computer, all following experiments were run on a 333Mhz Pentium II. When comparing absolute training times, one should keep in mind that SMO and Chunking were run on a slightly slower computer.

8.7.2.1 Income Prediction

This task was compiled by John Platt (see [Platt, 1998]) from the UCI “adult” data set. The goal is to predict whether a household has an income greater than $50,000. After discretization of the continuous attributes, there are 123 binary features. On average, there are ≈14 non-zero attributes per example.

Using the same parameters as in [Platt, 1998], Table 8.4 and the left-hand graph in Figure 8.2 show training times for the linear SVM with $C = 0.05$. Similarly, Table 8.5 and the right-hand graph in Figure 8.2 show training times for an RBF-kernel (3.34) with $\gamma = 0.05$ and $C = 1$. The results for SMO and Chunking are taken from [Platt, 1998].

In terms of absolute runtime, both SVM^{light} and SMO are substantially faster than the conventional chunking algorithm. For the RBF-kernel SVM^{light} is more than three times faster than SMO. The best working-set size is $q = 10$. In the linear case SVM^{light} outperforms SMO for large data sets. The best working-set size is $q = 20$.

By fitting lines to the log-log plot it is possible to get an empirical scaling of processing time vs. the number of training examples. In the linear case SVM^{light} shows a much better scaling of $n^{1.5}$ compared to SMO with $n^{1.9}$. This advantage is due to shrinking and the larger working-set size. While SMO repeatedly iterates over all support vectors, shrinking quickly identifies and removes the BSVs early in the optimization process. In addition, the increased working-set size leads to fewer iterations. For the RBF-kernel the scaling is $n^{2.1}$ for both SVM^{light} and SMO. The scaling of the chunking algorithm is much worse in both cases.

Examples	SVM^{light}	SMO	Chunking	Minimum	total SV	BSV
1605	0.6	0.4	37.1	$<$0.01	675	633
2265	1.0	0.9	228.3	$<$0.01	978	930
3185	2.1	1.8	596.2	$<$0.01	1268	1210
4781	2.9	3.6	1954.2	0.01	1856	1793
6414	4.1	5.5	3684.6	0.01	2433	2368
11221	9.5	17.0	20711.3	0.03	4158	4083
16101	19.1	35.3	N/A	0.04	5928	5854
22697	35.7	85.7	N/A	0.05	8326	8203
32562	59.4	163.6	N/A	0.07	11708	11542
Scaling	1.5	1.9	3.1	1.0		

Table 8.4: Training times and number of SVs for the income prediction data with linear kernel.

Examples	SVM^{light}	SMO	Chunking	Minimum	total SV	BSV
1605	3.1	15.8	34.8	1.6	691	584
2265	6.4	32.1	144.7	3.5	1007	849
3185	12.4	66.2	380.5	6.3	1297	1112
4781	28.9	146.6	1137.2	14.1	1888	1651
6414	52.8	258.8	2530.6	30.3	2488	2180
11221	205.6	781.4	11910.6	97.4	4221	3731
16101	433.1	1784.4	N/A	212.0	5948	5344
22697	881.6	4126.4	N/A	425.1	8346	7502
32562	1829.7	7749.6	N/A	857.4	11690	10604
Scaling	2.1	2.1	2.9	2.0		

Table 8.5: Training times and number of SVs for the income prediction data using an RBF-kernel

The column “minimum” gives a lower bound on the training time. It is the training time of an imaginary “oracle” algorithm that could guess the solution of the optimization problem and simply needed to verify that the guess was correct.

For the non-linear case the bound makes two assumptions. It assumes that there is no other solution of the optimization problem with substantially fewer support vectors, and that any optimization algorithms needs to at least once look at the rows of the Hessian Q which correspond to the support vectors. The column “minimum” shows the time to compute those rows once (exploiting symmetry). This time scales with $n^{2.0}$, showing the complexity inherent in the classification task. For the training set sizes considered, SVM^{light} is both close to this minimum scaling as well as within a factor of approximately two in terms of absolute runtime. Therefore, if a particular Hessian Q does not show regularities that could be exploited for an efficient approximation, the maximum speedup that could be achieved by an improved algorithm is small.

In the linear case the “oracle” algorithm can verify its guess much more efficiently. Kernel computations can be avoided by using a single weight vector. Here, SVM^{light} is much further away from the optimum, still leaving much room for improvements.

Examples	SVM^{light}	SMO	Chunking	Minimum	total SV	BSV
2477	3.2	2.2	13.1	<0.01	173	47
3470	6.5	4.9	16.1	<0.01	230	71
4912	8.5	8.1	40.6	0.01	277	106
7366	18.8	12.7	140.7	0.02	380	163
9888	24.7	24.7	239.3	0.03	471	240
17188	44.5	65.4	1633.3	0.04	753	477
24692	60.0	104.9	3369.7	0.05	991	682
49749	126.7	268.3	17164.7	0.09	1755	1396
Scaling	1.2	1.6	2.5	1.0		

Table 8.6: Training times and number of SVs for the Web data with linear kernel.

Examples	SVM^{light}	SMO	Chunking	Minimum	total SV	BSV
2477	7.5	26.3	64.9	1.3	430	47
3470	13.4	44.1	110.4	2.6	575	69
4912	21.2	83.6	372.5	4.5	675	96
7366	49.4	156.7	545.4	10.3	882	138
9888	70.8	248.1	907.6	15.9	1076	187
17188	212.5	581.0	3317.9	46.8	1607	364
24692	402.1	1214.0	6659.7	78.4	1997	506
49749	1458.8	3863.5	23877.6	298.7	3081	949
Scaling	1.7	1.7	2.0	1.7		

Table 8.7: Training times and number of SVs for the Web data using an RBF-kernel.

8.7.2.2 Classifying Web Pages

The second data set - again compiled by John Platt (see [Platt, 1998]) - is a text-classification problem with a binary representation based on 300 keyword features. This representation is extremely sparse. On average there are only ≈12 non-zero features per example.

Tables 8.6 and 8.7 show the training times on this data set for the linear SVM with $C = 1$ and an RBF-kernel with $\gamma = 0.05$ and $C = 5$. The training times are also plotted as graphs in Figure 8.3. Again, the times for SMO and Chunking are taken from [Platt, 1998].

For the RBF-kernel SVM^{light} is faster than SMO and Chunking on this data set as well, scaling with $n^{1.7}$. The best working-set size is $q = 2$. In the linear case SVM^{light} outperforms SMO with increasing training set size. Here, the best working-set size is $q = 20$. Again, the scaling of SVM^{light} ($n^{1.2}$) is substantially better than that of SMO ($n^{1.6}$) and Chunking ($n^{2.5}$).

8.7.2.3 Extended Ohsumed Data Set

The previous experiments using the Ohsumed data set used only the first 10,000 documents. Now, the training set size is extended to up to the first 46,160 documents from

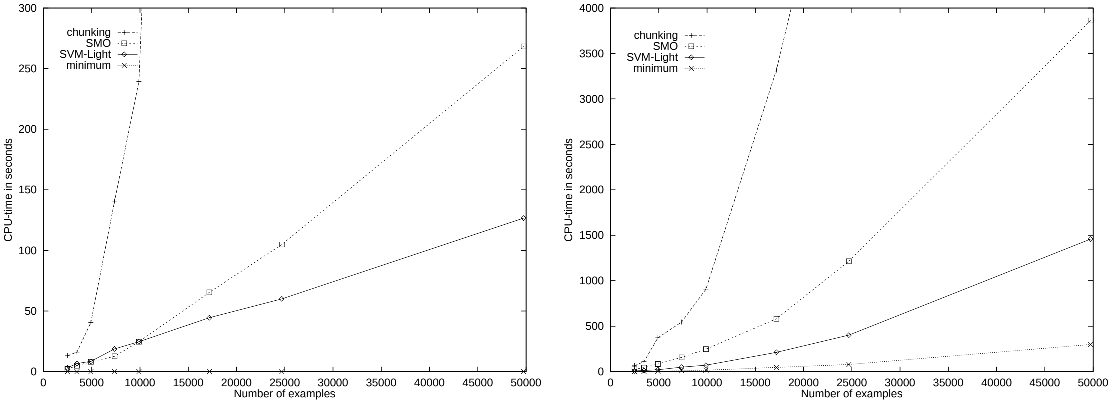


Figure 8.3: Training times for the linear SVM (left) and the RBF-kernel (right) on the Web page classification task. The times are repeated from tables 8.6 and 8.5.

Examples	SVM^{light}	Minimum	total SV	BSV
9337	10.0	0.09	2662	1319
13835	19.2	0.12	3507	1852
27774	56.3	0.25	5687	3361
46160	115.8	0.40	8343	5426
Scaling	1.5	1.0		

Table 8.8: Training time and number of SVs for different training set sizes of the Ohsumed data with linear kernel.

Examples	SVM^{light}	Minimum	total SV	BSV
9337	617.5	278.6	4037	0
13835	1583.3	542.5	5384	0
27774	6005.2	1840.2	9016	0
46160	16471.3	4769.4	13811	0
Scaling	2.0	1.8		

Table 8.9: Training time and number of SVs for different training set sizes of the Ohsumed data using an RBF-kernel.

1991. The particular task is to learn the “Cardiovascular Diseases” category. On average, there are ≈63 non-zero features per example.

Table 8.8 shows the training times of the linear SVM with $C = 0.5$. Table 8.9 gives the training times for an RBF-kernel with $\gamma = 0.6$ and $C = 50$. The RBF-kernel leads to many SVs which are not at the upper bound. Relative to this high number of SVs the cache size is small. To avoid frequent recomputations of the same part of the Hessian Q, an additional heuristic is incorporated here. The working set is selected with the constraint that for at least half of the selected variables the kernel values are already cached. Optimum performance is achieved with a working-set size of $q = 20$ and $h = 400$. For the training set sizes considered here, runtime is within a factor of 4 from the minimum. The scaling in the linear case ($n^{1.5}$) is again much better than for the non-linear kernel ($n^{2.0}$). The best working-set size in the linear case is $q = 30$.

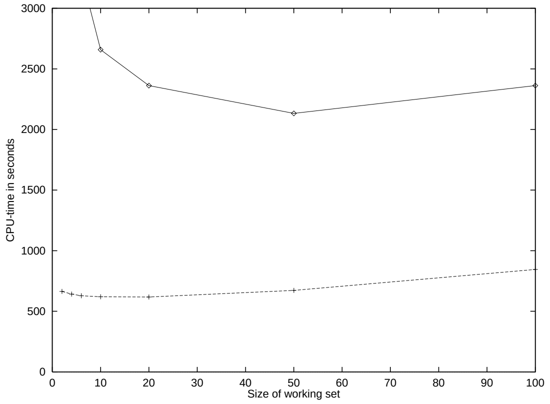


Figure 8.4: Training time dependent on working-set size for the Ohsumed task.

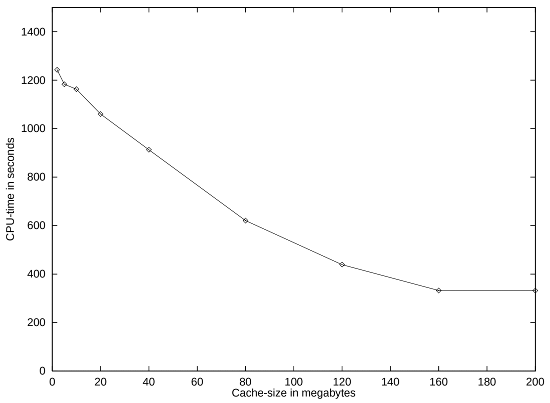


Figure 8.5: Training time dependent on cache size for the Ohsumed task.

Let us now evaluate how particular strategies of the algorithm influence the performance.

8.7.3 What is the Influence of the Working-Set-Selection Strategy?

Figure 8.4 plots training time dependent on the size q of the working set. It shows the results for the smallest training set of the extended Ohsumed task using the RBF-kernel. The selection strategy from Section 8.3 (lower curve) is compared to a basic strategy similar to that proposed in [Osuna et al., 1996] (upper curve). In each iteration the basic strategy randomly replaces half of the working set with variables that do not fulfill the optimality conditions. The graph shows that the new selection strategy reduces time by a factor of more than 3.

8.7.4 What is the Influence of Caching?

Figure 8.5 shows that caching has a strong impact on training time. It plots training time for the RBF-kernel on the 9,337 examples of the Ohsumed data dependent on the cache size. With the cache size ranging from 2 megabytes to 200 megabytes a speedup factor of almost 4 is achieved. The speedup generally increases with an increasing density of the feature vectors $\vec{x}_i$.

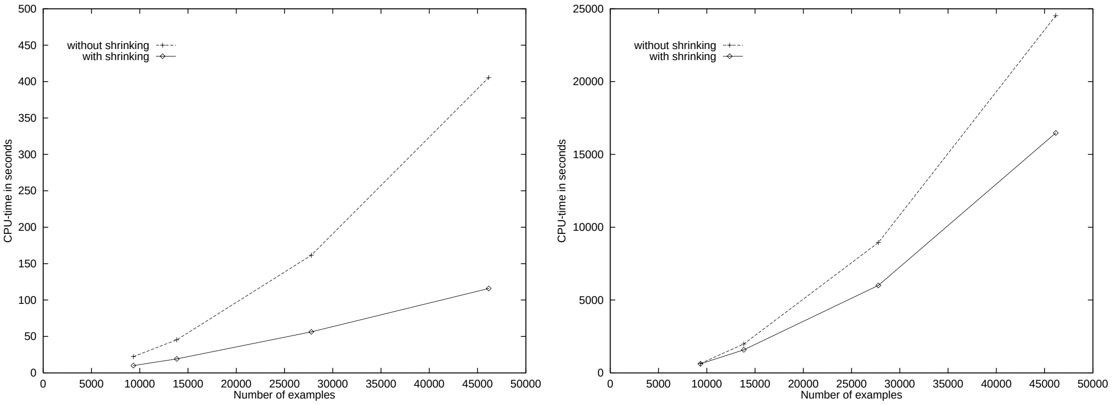


Figure 8.6: Training times with and without shrinking for the linear SVM (left) and the RBF-kernel (right) on the extended Ohsumed task.

8.7.5 What is the Influence of Shrinking?

All experiments above use the shrinking strategy from Section 8.4. Figure 8.6 (linear case left, RBF-kernel right) shows the training time with and without shrinking. For the linear SVM shrinking substantially reduces absolute training time and improves the scaling. While smaller, shrinking provides a speedup also for the RBF-kernel. For non-linear kernels the gain depends on the cache size in relation to the size of the problem and the number of support vectors. It is most effective when the cache is small compared to the size of the problem. In both the linear and the non-linear case, the gain of shrinking generally increases the smaller the fraction of unbounded SVs is compared to the number of training examples n.

8.8 Summary and Conclusions

This chaper presented an improved algorithm for training SVMs on large-scale problems. It is substantially more efficient than the conventional chunking algorithm with a projected conjugate gradient solver. The algorithm was the first to show that it is tractable to train SVMs also for large problems with many support vectors and many features. The algorithm is based on a decomposition strategy and includes the first principled approach to the problem of selecting the variables for the working set in an effective and efficient way. Furthermore, a technique for "shrinking" the problem during the optimization process is introduced. This is found particularly effective for large learning tasks where the fraction of SVs is small compared to the sample size, or when many SVs are at the upper bound.

The chapter also describes how this algorithm is efficiently implemented in SVM^{light} . It was the first SVM learner suitable for training text classifiers with reasonably sized training sets. Since the algorithm exploits the sparsity of feature vectors, it is particularly efficient for learning text classifiers. Using only dot products based on a sparse vector representation, the time complexity of an iteration is independent of the dimensionality of the feature space. The memory requirements are linear in the number of training examples and in the number of SVs. Nevertheless, the algorithms can benefit from additional storage space, since the caching strategy allows an elegant trade-off between training time and memory consumption.

Chapter 9

Training Transductive Support Vector Machines

Chapter 7 shows that a transductive approach to text classification can lead to improved predictive performance. Especially when the number of labeled training examples is small and the test set is large, a transductive SVM (TSVM) can offer a substantial benefit over an inductive SVM. However, the problem of computational efficiency in training transductive SVMs has not been considered yet.

This chapter tackles the optimization problem associated with training transductive SVMs. The optimization problem has the form of a mixed integer quadratic program. In general, this problem is not convex and the number of integer variables scales with the number of test examples. So finding the global optimum with known methods is possible only for very small test sets. Therefore, conventional optimization methods are not applicable to training transductive SVMs, since TSVMs are useful only when the test set is large.

The following proposes an efficient algorithm that finds an approximate solution even for large test sets. The algorithm is analyzed and shown to converge. Its runtime is evaluated on several text-classificaton tasks. While it takes longer to train than an inductive SVM, the algorithm makes the transductive approach tractable even for large test sets with several thousands of examples.

The algorithm is publicly available as part of SVM^{light} V3.10.

9.1 Problem and Approach

Chapter 7 shows based on [Vapnik, 1998] how training a transductive SVM leads to the following (partly) combinatorial optimization problem.

Optimization Problem 14 (Transductive SVM (non-separable case))

$$\begin{array}{ll} \textit{minimize:} & V(y_1^*,...,y_k^*,\vec{w},b,\xi_1,...,\xi_n,\xi_1^*,...,\xi_k^*) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=0}^{n}\xi_i + C^*\sum_{j=0}^{k}\xi_j^* \\ \textit{subject to:} & \forall_{i=1}^{n}: y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i \\ & \forall_{j=1}^{k}: y_j^*[\vec{w}\cdot\vec{x}_j^* + b] \geq 1 - \xi_j^* \\ & \forall_{i=1}^{n}: \xi_i > 0 \\ & \forall_{j=1}^{k}: \xi_j^* > 0 \\ & \forall_{j=1}^{k}: y_j^* \in \{-1,+1\} \end{array}$$

For a small number of test examples, this problem can be solved optimally simply by trying all possible assignments of $y_1^*, ..., y_k^*$ to the two classes. However, this approach become intractable for test sets with more than 10 examples. Previous approaches using branch-and-bound search [Vapnik and Sterin, 1977, Wapnik and Tscherwonenkis, 1979] push the limit to some extent, but their maximum test set size is still limited too less than 100 test examples. Since the transductive SVMs were found to be most useful for large test sets, these algorithms are not appropriate for learning text classifiers.

The algorithm proposed next is designed to handle the large test sets common in text classification with 10,000 test examples and more. It finds an approximate solution to Optimization Problem 14 using a form of local search. Local search algorithms start with some initial instantiation of the variables. In each iteration the current variable instantiation is modified so that it moves closer to a solution. This process is iterated until no further improvement is possible.

At the first glance, a straightforward way to apply this algorithm to solving OP14 is the following. Start with some labeling of the test examples. Then switch the label of some example in each iteration, so that the size of the margin increases. Nevertheless, at the second glance there are two problems with this approach. First, there is no obvious sufficient criterion for selecting which label to switch in each iteration. Finding an example so that switching its class increases the margin requires actually performing the switching and retraining. This makes the selection process very expensive. Second, there are many local minima so that the search gets stuck before much progress was made.

The algorithm presented here circumvents these problems by using a smooth approximation to the objective function of OP14. This approximation is iteratively made closer and closer as the optimization proceeds. Eventually, the approximation is identical to the objective function of OP14. For the approximated objective function there is a suitable sufficient condition for improvement. In addition, the algorithm was empirically found to be robust against getting stuck in local minima far away from the optimum.

9.2 The TSVM Algorithm

The algorithm takes the training data and the test examples as input and outputs the predicted classification of the test examples. Besides the two parameters C and C^*, the user can specify the number of test examples to be assigned to class $+$. This allows trading-off recall vs. precision. The following description of the algorithm covers only the linear case. A generalization to non-linear hypothesis spaces using kernels is straightforward.

The algorithm is summarized in Figure 9.1. It starts with training an inductive SVM on the training data. The initial labeling of test data is based on the classification of this inductive SVM. While the inductive SVM providing the initial labeling does not take the test examples into account, Loop 1 uniformly increases the influence of the test examples by incrementing the cost-factors C_-^* and C_+^* up to the user-defined value of C^*. This can be viewed as a slow shift between a fully inductive ($C_-^* = C_+^* = 0$) and a fully transductive SVM ($C_-^* = C_+^* = C^*$).

At each level of approximation, Loop 2 iteratively improves the solution by switching the labels of a pair of test examples. The criterion in the condition of Loop 2 identifies two examples for which changing the class labels leads to a decrease in the current objective function. If there are no more such examples, the algorithm moves to a closer (but less smooth) approximation in Loop 1. The algorithm uses unbalanced costs C_-^* and C_+^* to better accomodate the user-defined ratio num_+.

```
Algorithm TSVM:

Input:        – training examples $(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)$
              – test examples $\vec{x}^*_1, ..., \vec{x}^*_k$
Parameters:   – $C$,$C^*$: parameters from OP14
              – $num_+$: number of test examples to be assigned to class +
Output:       – predicted labels of the test examples $y^*_1, ..., y^*_k$


$(\vec{w}, b, \vec{\xi}, \_) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [], C, 0, 0)$;

Classify the test examples using $< \vec{w}, b >$. The $num_+$ test examples with
the highest value of $\vec{w} * \vec{x}^*_j + b$ are assigned to the class $+$ ($y^*_j := 1$); the
remaining test examples are assigned to class $-$ ($y^*_j := -1$).

$C^*_- := 10^{-5}$;                       // some small number
$C^*_+ := 10^{-5} * \frac{num_+}{k - num_+}$;

while(($C^*_- < C^*$) || ($C^*_+ < C^*$)){                                 // Loop 1

    $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [(\vec{x}^*_1, y^*_1)...(\vec{x}^*_k, y^*_k)], C, C^*_-, C^*_+)$;

    while($\exists m, l : (y^*_m * y^*_l < 0) \& (\xi^*_m > 0) \& (\xi^*_l > 0) \& (\xi^*_m + \xi^*_l > 2)$) {     // Loop 2

        $y^*_m := -y^*_m$;                // take a positive and a negative test
        $y^*_l := -y^*_l$;                // example, switch their labels, and retrain

        $(\vec{w}, b, \vec{\xi}, \vec{\xi}^*) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [(\vec{x}^*_1, y^*_1)...(\vec{x}^*_k, y^*_k)], C, C^*_-, C^*_+)$;

    }

    $C^*_- := min(C^*_- * 2, C^*)$;
    $C^*_+ := min(C^*_+ * 2, C^*)$;

}

return($y^*_1, ..., y^*_k$);
```

Figure 9.1: Algorithm for training transductive support vector machines.

The function *solve_svm_qp* is used throughout the algorithm as a sub-procedure. It refers to quadratic programs of the following type.

Optimization Problem 15 (Inductive SVM (primal))

$$\textit{minimize:} \quad V(\vec{w}, b, \vec{\xi}, \vec{\xi}^*) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n}\xi_i + C_-^*\sum_{j:y_j^*=-1}\xi_j^* + C_+^*\sum_{j:y_j^*=1}\xi_j^*$$

$$\begin{aligned} \textit{subject to:} \quad & \forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i \\ & \forall_{j=1}^{k} : y_j^*[\vec{w}\cdot\vec{x}_j + b] \geq 1 - \xi_j^* \\ & \forall_{i=1}^{n} : \xi_i > 0 \\ & \forall_{j=1}^{k} : \xi_j^* > 0 \end{aligned}$$

This optimization problem is similar to an inductive SVM. It is solved in its dual formulation using the algorithm described in Chapter 8. For the calls to *solve_svm_qp* from within Loop 1 and Loop 2, the optimizer uses the current solution as a starting point. For the call withing Loop 2 the current optimization problem differs from the previous only with respect to the labels of two examples. So the solutions of both problems are generally similar and starting the optimizer at the old solution greatly reduces runtime. The same argument also applies to the call of *solve_svm_qp* at the beginning of Loop 1. Here, only the upper bounds C_-^* and C_+^* differentiate it from the previous optimization problem.

9.3 Analysis of the Algorithm

This section first gives an intuitive mechanical interpretation of the algorithm before it analyses the algorithm formally.

9.3.1 How does the Algorithm work?

The basic intuition behind the TSVM algorithm can be best explained using an analogy for the SVMs solution in terms of forces and torgues. In [Burges, 1998] it is identified that the SVM solution has a mechanical interpretation. At the solution, the optimal hyperplane is held by a mechanical system that applies a perpendicular force of α_i to the hyperplane form each training example i. This means that each support vector pushes against the hyperplane with force α_i, keeping it in a stable position.

The example in Figure 9.2 shows how this analogy applies to the TSVM algorithm. Each plus sign indicates a positive training example, minus signs indicate negative training examples, and dots stand for test examples. In a) the solution of an inductive SVM is depicted. The shaded area indicates the margin. The test examples are classified according to the inductive SVM and the algorithm enters Loop 1. By increasing the values of C_-^* and C_+^*, the test examples are now allowed to apply a small force to the hyperplane as well. More specific, those test examples that lie within the margin now apply a force pulling at the margin boundaries as indicated by the lines. In the example, this "rotates" the hyperplane counter-clockwise as indicated in b).

The rotation continues with the allowed force increasing in Loop 1 until one of two cases occurs. First, examples can leave the margin area as indicated in the lower right corner of b). Nevertheless, for the example in Figure 9.2, the hyperplane keeps moving with increasing force. The second case is that at least one test example from each class lies beyond a line parallel to the hyperplane. This is the case in c). When this happens, the condition of Loop 2 is fulfilled and the labels of those test examples are switched.

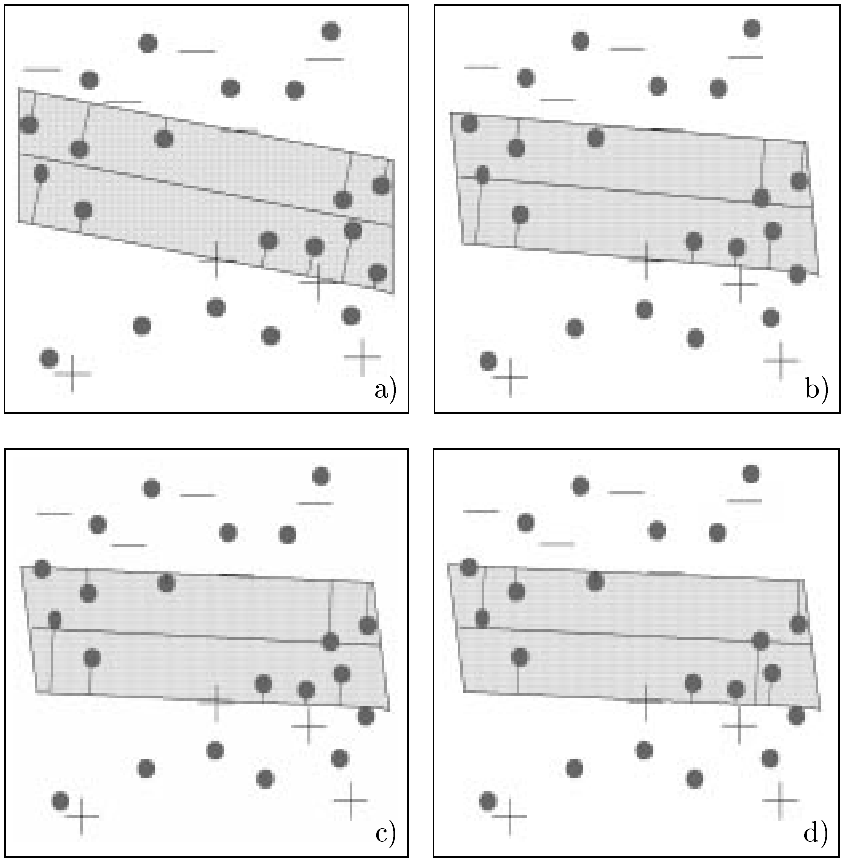


Figure 9.2: Intuitive example for the TSVM algorithm.

After this switch, d) depicts how this changes the direction of the force. Those examples now pull to the opposite direction. The process iterates until there are no examples left in the margin or until the upper bound C^* is reached.

9.3.2 Convergence

What are the algorithmic properties of the algorithm? The following theorem shows that the algorithm does not cycle and converges in a finite number of steps.

Theorem 15 *Algorithm 1 converges in a finite number of steps.*

Proof *To prove this, it is necessary to show that Loop 2 is exited after a finite number of iterations. This holds since the objective function of optimization problem OP10 decreases with every iteration of Loop 2 as the following argument shows. The condition $y_m^* y_l^* < 0$ in Loop 2 requires that the examples to be switched have different class labels. Assign $\xi_m^{*'} = max(2 - \xi_m^*, 0)$ and $\xi_l^{*'} = max(2 - \xi_l^*, 0)$. Let $y_m^* = 1$ so that we can write*

$$\frac{1}{2}\vec{w} \cdot \vec{w} + C\sum_{i=0}^{n} \xi_i + C_-^* \sum_{j:y_j^*=-1} \xi_i^* + C_+^* \sum_{j:y_j^*=1} \xi_i^*$$

$$= \frac{1}{2}\vec{w} \cdot \vec{w} + C\sum_{i=0}^{n} \xi_i + ... + C_+^* \xi_m^* + ... + C_-^* \xi_l^* + ...$$

$$> \frac{1}{2}\vec{w}\cdot\vec{w}+C\sum_{i=0}^{n}\xi_i + ... + C_-^*\xi_m^{*'} + ... + C_+^*\xi_l^{*'} + ...$$

The inequality holds due to the selection criterion $\xi_m^* + \xi_l^* > 2, \xi_m^* > 0, \xi_l^* > 0$ in Loop 2, so that $\xi_m^{*'} = max(2-\xi_m^*, 0) < \xi_l^*$ and $\xi_l^{*'} = max(2-\xi_l^*, 0) < \xi_m^*$. It remains to be verified that the constraints of OP10 are fulfilled for the new values $y_m^{*'}$, $y_l^{*'}$, $\xi_m^{*'}$, and $\xi_l^{*'}$. It is clear from the construction that the positivity constraints for $\xi_m^{*'}$ and $\xi_l^{*'}$ are fulfilled. The following shows that the constraints $y_j^{*'}[\vec{w}\cdot\vec{x}_j + b] \geq 1 - \xi_j^{*'}, j \in \{l, m\}$ are also fulfilled. For the old values it holds that $y_j^*[\vec{w}\cdot\vec{x}_j + b] = 1 - \xi_j^*$, since $\xi_j^* > 0$ is required in the condition of Loop 2. This implies

$$\begin{aligned}
1 - \xi_j^{*'} &= 1 - max(2 - \xi_j^*, 0) \\
&\leq -1 + \xi_j^* \\
&= -y_j^*[\vec{w}\cdot\vec{x}_j + b] \\
&= y_j^{*'}[\vec{w}\cdot\vec{x}_j + b]
\end{aligned}$$

This completes the argument that the objective function decreases with every step. It follows that Loop 2 is exited after a finite number of iterations, since there is only a finite number of permutations of the test examples. Loop 1 also terminates after a finite number of iterations, since C_-^* is bounded by C^*. ∎

While convergence is an important property, the theorem does not give much information about the rate of convergence. Surely, the algorithm cannot make more than $O(2^k)$ switches. Each switch provides a strict descent in the objective function and there are only 2^k possible states for each approximation. Nevertheless, this bound is not very useful in practice. Finding a tighter bound is an open problem. Such a bound is likely to exist, since the following experiments show that the number of switches is generally very small.

Furthermore, the theorem does not describe the quality of the solution found. It is an open problem to characterize situations in which the algorithm converges to the global optimum. That the algorithm does effectively maximize margin is shown experimentally.

9.4 Experiments

The experiments in this section evaluate the quality and the efficiency of the algorithm. The first experiment briefly verifies that the algorithm does in fact maximize margin as expected. The main focus is on the evaluation of its efficiency. If not noted otherwise, the runtimes are averages over multiple test/training splits for a linear TSVM with $C = 20$. They correspond to the experiments from Chapter 7. The cpu-times are measured on a SUN Ultra 10 with a 300Mhz CPU.

9.4.1 Does the Algorithm Effectively Maximize Margin?

The plot in Figure 9.3 verifies that the algorithm does maximize margin. It shows an experiment for the Reuters category 'acq' using 5 training and 300 test examples from each class. The x-axis of the plot shows the number of iterations through Loop 2 or equivalently the number of switches. On the y-axis the normalized inverse margin $\frac{R^2}{\delta^2}$ is plotted, if the algorithm was stopped after the n-th iteration. The plot shows that the TSVM algorithm does almost uniformly improve the labeling of the test data so that the margin is increased by a factor of almost two.

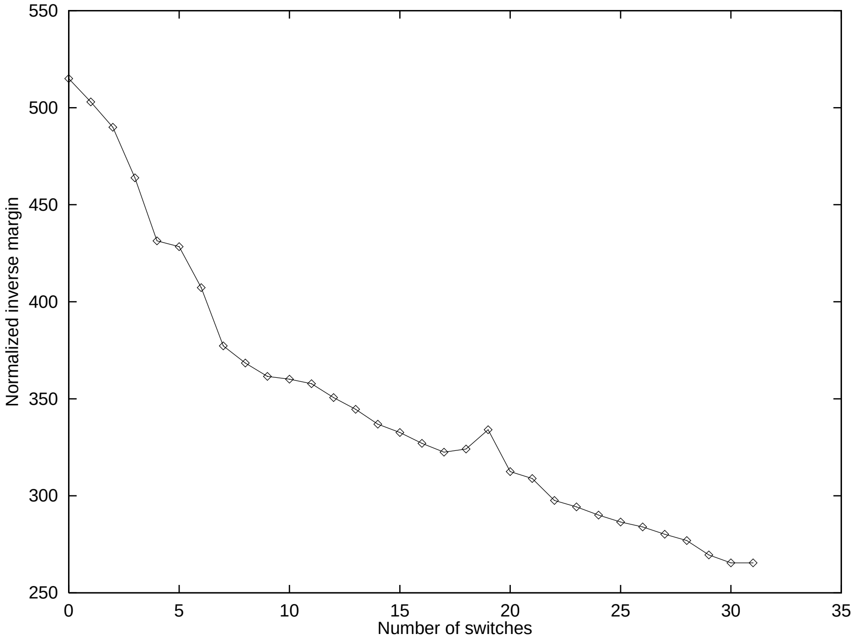


Figure 9.3: The inverse relative margin after a given number of iterations of the TSVM algorithm.

9.4.2 Training Times for Reuters, WebKB, and Ohsumed

This section evaluates the training times of the TSVM algorithm on text-classification tasks. Tables 9.1, 9.2, and 9.3 give the training times for the ten most frequent Reuters categories, the WebKB data set, and the five most frequent Ohsumed categories. The training times are on the order of minutes and are substantially longer than those for an inductive SVM. For the small training set sizes considered here, inductive SVMs lie well below one second on all categories. However, the TSVM algorithm is substantially faster than, for example, conventional branch-and-bound methods. They are intractable for the size of problem considered here.

The tables also show the number of switches that are performed in Loop 2. It is relatively small and far away from a potential worst case of 2^k. The following section shows how the number of switches relates to training time.

9.4.3 How does Training Time Scale with the Number of Training Examples?

While for inductive learning algorithms training time is expected to increase with the number of training examples, this is not necessarily the case for transductive learning algorithms. Figure 9.4 plots training time depending on the number of training examples. Beginning with very small training sets, time decreases with an increasing training set size. Only for larger training sets the trend changes again. This can be explained as follows. The figure also plots the number of iterations in Loop 2. For small training sets the TSVM algorithm changes the labels of many test examples so that Loop 2 is repeated often. For larger training sets the labeling of the test examples produced by the inductive SVM is already close to the transductive solution. This leads to fewer iterations in Loop 2 and therefore lower training time. On the other hand, each call to *solve_svm_qp* takes longer for a larger training set. So for very large training sets the additional cost of *solve_svm_qp* outweighs the savings from a reduced number of switches, and overall

	Avg. Training Time (cpu-seconds)	Avg. number of Switches
earn	131.5	271
acq	138.9	554
money-fx	104.6	146
grain	106.9	87
crude	110.1	256
trade	98.5	137
interest	99.1	119
ship	97.3	130
wheat	89.2	59
corn	87.0	41

Table 9.1: Average training time and average number of switches for the Reuters data with 17 training examples and 3,299 test examples.

	Avg. Training Time (cpu-seconds)	Avg. number of Switches
course	449.9	501
faculty	500.9	1065
project	427.3	878
student	513.7	850

Table 9.2: Average training time and average number of switches for the WebKB data with 9 training examples and 3,957 test examples.

	Avg. Training Time (cpu-seconds)	Avg. number of Switches
Pathology	2041.1	1833
Cardiovascular	1667.0	807
Neoplasms	1647.0	884
Nervous System	1766.3	1248
Immunologic	1497.6	838

Table 9.3: Average training time and average number of switches for the Ohsumed data with 120 training examples and 10,000 test examples.

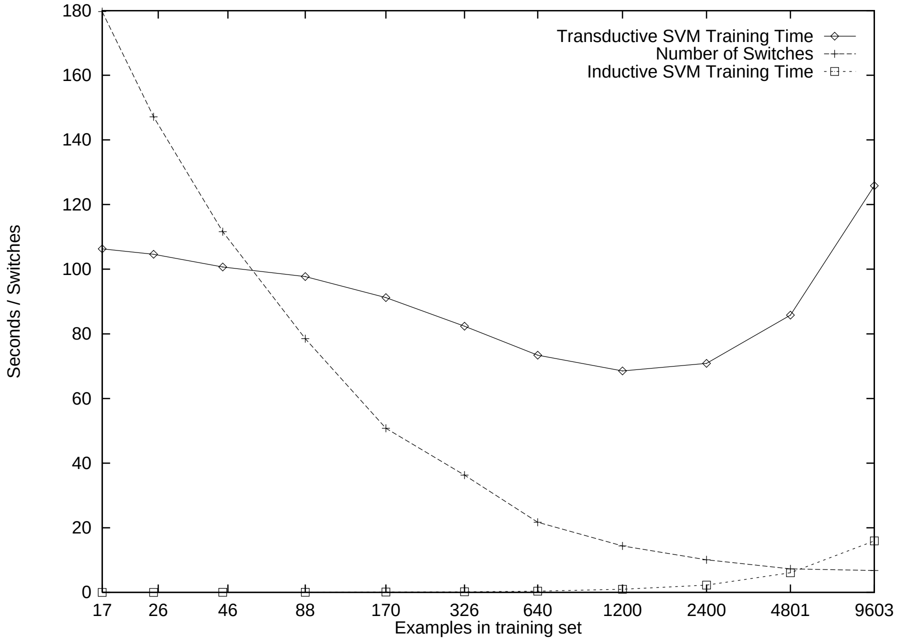


Figure 9.4: Training time and number of switches averaged over the ten most frequent Reuters categories as a function of the training set size. The test set size is 3299 examples.

training time increases.

Figure 9.4 also shows the average training time of an inductive linear SVM with $C = 20$. The inductive SVM is faster than the TSVM for all training set sizes. Since the inductive SVM is independent of the test set, the difference is largest for small training sets.

9.4.4 How does Training Time Scale with the Number of Test Examples?

Transductive learning depends not only on the training set, about also on the test set. So it is necessary to ask how training time scales with the number of test examples. Figure 9.5 shows training time and number of switches depending on the number of test examples. Both increase superlinearly with the size of the test set. The scaling of the training time is approximately $k^{1.5}$.

9.5 Related Work

The first training algorithm for transductive support vector machines was proposed in [Vapnik and Sterin, 1977] and refined in [Wapnik and Tscherwonenkis, 1979]. It is based on a branch-and-bound algorithm. While it finds the optimal solution, the authors already recognized that it applies only to problems with small test sets. They present experiments with up to 60 test examples. For larger scale problems, Vapnik and Tschervonenkis mention local search algorithms as a potential solution [Wapnik and Tscherwonenkis, 1979]. The algorithm they sketch is similar to the naive local search algorithm from Section 9.2.

After a long period with no activity, Bennett and Demiriz recently proposed another approach [Bennett and Demiriz, 1998, Bennett, 1999]. They consider a modified problem

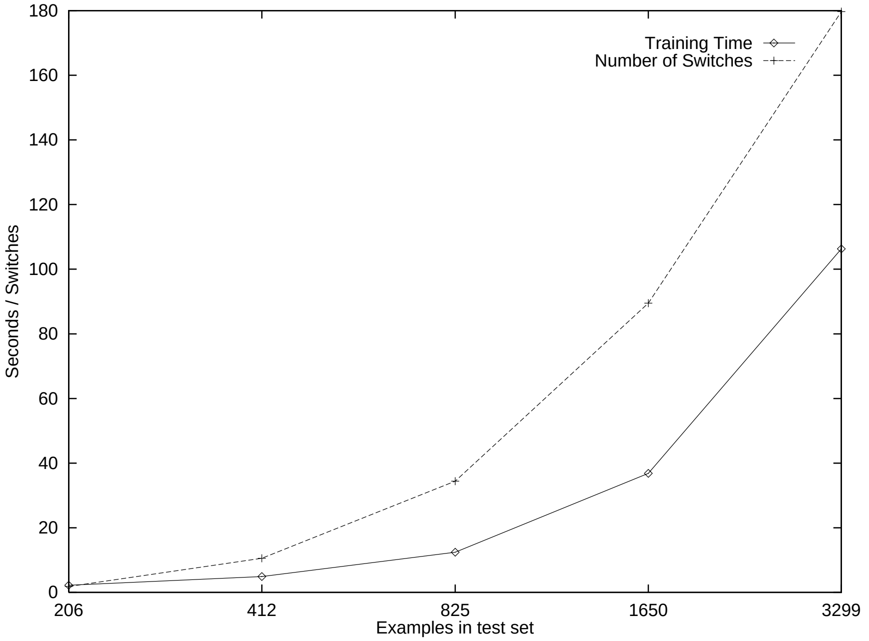


Figure 9.5: Training time and number of switches averaged over the ten most frequent Reuters categories as a function of the test set size. The training set consists of 17 examples.

that is easier to solve. Instead of using the 2-norm of the weight vector in the objective function of the SVM, they replace it with the 1-norm. While this results in a different learning algorithm, they show that this problem can be transformed into a (linear) mixed integer program. This problem can be solved using the standard CPLEX optimization software. Nevertheless, even for this modified formulation the maximum test set size is around 70 depending on the particular branching factor.

Another approach using the modified problem proposed by Bennett and Demiriz is currently explored by Fung and Mangasarian [Fung and Mangasarian, 1999] . They avoid the mixed integer formulation and instead use an approach based on repeated linear programming. Similar to the algorithm presented here, they drop the requirement of having to find the global optimum for the benefit of improved efficiency. Their repeated linear programming algorithm can handle test sets much larger than mixed integer programming. Nevertheless, their approach does not apply to regular support vector machines considered in this chapter and throughout this dissertation. It is an open question whether the modified formulation provides good learning results for text classification.

9.6 Summary and Conclusions

This chapter proposes an algorithm for training transductive support vector machines efficiently. It is the first algorithm that can handle test sets with more than 100 examples. The algorithm proceeds by repeatedly optimizing closer and closer approximations to the TSVM training problem using local search. The algorithm is analyzed and shown to converge in a finite number of steps. The solutions found by the algorithm are evaluated on text-classification tasks and shown to lead to substantial improvements in generalization performance.

The efficiency of the algorithm is evaluated experimentally. Compared to training an inductive SVM, the TSVM algorithm requires a substantially longer training time. The relative increase is largest when the transductive approach shows the largest benefits – namely for small training sets and large test sets. Nevertheless, training is still tractable even for large test sets with thousands of examples.

Chapter 10

Conclusions

The contributions of this dissertation are threefold. From an application perspective, it provides a new approach to solving the problem of learning text classifiers from examples that overcomes the limitations of conventional methods. From a technical perspective, this dissertation develops new methods in the field of support vector learning that are independent of the application domain and can apply to a large range of other machine-learning tasks. And finally, on a meta level, this dissertation as a whole serves as a case study for how to approach and analyze a class of high-dimensional learning problems that can be transferred to other domains. These three views on this dissertation are discussed separately in the following.

10.1 Conclusions and Contributions on the Application Level

This dissertation presents a new machine-learning approach to the problem of learning text classifiers from examples. It is not primarily about methods, nor primarily about theory, nor primarily about algorithms. Rather, it addressed all relevant aspects of this particular class of learning problems. It is the first approach to learning text classifiers from examples

- that is computationally efficient,
- for which there is a justified learning theory that describes its mechanics with respect to text classification, and
- that performs well and robustly in practice.

No conventional method has provided a solution that covers all three points sufficiently. Generative modeling based methods, like naive Bayes, make unreasonable assumptions about text-classification tasks and show suboptimal prediction accuracy. Non-parametric classifiers, like k-NN, show better performance, but lack a theoretical learning model applicable to text classification. Like most other learning methods (e.g. decision tree classifiers, neural nets, etc.) they depend on the dimensionality of the feature space, which makes them — without further justification — inappropriate for learning text classifiers.

The key insight that drives the methods, the theory, and the algorithms developed in this dissertation is a new complexity measure for text classification. It is based on the idea of maximizing margin as developed in statistical learning theory. This dissertation shows that a maximum-margin approach to learning text classifiers based on support vector machines provides five main advances on the application level:

Good Empirical Performance: Support vector machines provide state-of-the-art generalization performance, outperforming conventional learning methods substantially.

Appropriate Modeling of Text-Classification Tasks: Chapter 7 proposes a new framework to model many text classification and information retrieval tasks more appropriately. The new model is based on the transductive setting. In contrast to inductive learning, in the transductive setting the examples to be classified are part of the input to the learner. The maximum-margin approach presented in this dissertation offers the flexibility to model both scenarios. Each text-classification problem can therefore be modeled in the most appropriate way, exploiting all information that is available. The transductive model is novel to the fields of text classification and information retrieval. The benefit of the transductive approach was found to be most pronounced for learning tasks with little training data and large test sets.

Autonomy and Flexibility: The most appropriate document representation depends on the learning task. For conventional learning methods evaluating multiple representations is a time-consuming process that involves resampling methods like cross-validation or bootstrap. Chapter 6 shows how selecting among multiple representations, processing steps like stemming, stopword removal, and weighting schemes, as well as setting other learning parameters can be done efficiently and without need for expert interventions. In addition, the methods proposed in Chapter 5 deliver an estimate of the generalization performance without additional data or additional training time. The estimate makes it possible to automatically detect when the accuracy of a learned classification rule is appropriate for the task at hand. This makes the approach presented in this dissertation suitable even for situations where expert interventions are not possible (e.g. desktop applications).

Theoretical Validity: For none of the conventional methods is there an appropriate model that explains why and when they will perform well on a particular text-classification task. While the models for some methods, like the naive Bayes classifier, are overly restrictive and inappropriate for text, others, like decision tree learners, rely purely on empirical evidence. Their suitability for learning text classifiers is not well understood. The statistical learning model presented in Chapter 4 overcomes these deficiencies. It is the first model that connects the statistical properties of text-classification tasks with the generalization performance of a learning algorithm – here the support vector machine. The model explains how the maximum-margin approach can avoid the “curse of dimensionality” for text classification even without a feature-selection step that is essential for many conventional methods. This makes support vector machines the only learning method for which it is well understood when and why it works well for text classification. The model identifies sufficient conditions of text-classification tasks that provably lead to low classification error. Furthermore, for the first time it provides a justified formal model of text-classification tasks, making them accessible for theoretical analysis also with respect to other learning methods.

Computational Efficiency: Methods are of little practical use without efficient algorithms. In particular, the training phase of a learning algorithm can be inefficient. For the learning methods explored in this dissertation, chapters 8 and 9 propose efficient training algorithms. In contrast to most other learning methods, these algorithms do not necessarily depend on the dimensionality of the feature space, which makes them particularly appropriate for learning text classifiers.

Based on the original publications of the work presented in this dissertation, parts of the approach presented here were already validated by other researchers as outlined in the individual chapters. It has already inspired several extensions and found entrance into commercial applications.

10.2 Conclusions and Contributions on the Technical Level

This work develops some widely applicable machine-learning techniques, but limits their discussion to the text-classification problem. However, both the particular techniques and the general approach taken here are not limited to text classification. While it was possible to use existing results to achieve some of these advances in the application domain, the development of new methods and theory was necessary in several areas. They range from questions in statistical learning theory to mathematical programming issues.

The main technical advances provided in this dissertation are:

EFFICIENT PERFORMANCE ESTIMATION: Chapter 5 proposes and analyzes efficient estimators of the error rate, the precision, the recall, and other performance measures for classification rules generated with a support vector machine. The estimators can be computed at essentially no extra cost directly after training the SVM on the training set. This makes the new estimators an order of magnitude more efficient than naive cross-validation or bootstrap. In addition, Chapter 5 proposes how exact leave-one-out estimation can be sped up, making it practical even for large training sets.

EFFICIENT MODEL SELECTION: Chapter 6 shows how these estimators can be used for model selection regarding the input representation, the kernel, and the regularization parameter of a support vector machine.

UPPER BOUND FOR THE EXPECTED ERROR RATE OF AN SVM: Chapter 4 provides a new bound on the expected error of a support vector machine. It overcomes the problems of Vapnik's bound that applies only to the special case of unbiased support vector machines and that cannot handle inseparable training data.

EFFICIENT TRAINING ALGORITHM FOR INDUCTIVE SUPPORT VECTOR MACHINES: While training an inductive support vector machine can be reduced to a standard quadratic optimization problem, standard methods for solving such problems are inappropriate or even intractable for large data sets. Chapter 8 proposes and analyzes a new training algorithm for inductive SVMs that can handle large training sets with 50,000 and more examples efficiently.

TRANSDUCTIVE CLASSIFICATION: Chapter 7 analyzes and verifies when and why transductive learning algorithms provide an advantage over regular inductive support vector machines. It also develops criteria that help identify when the transductive predictions are likely to be suboptimal.

EFFICIENT TRAINING ALGORITHM FOR TRANSDUCTIVE SUPPORT VECTOR MACHINES: Training a support vector machine in the transductive framework requires solving a mixed integer quadratic program for which there is no known efficient solution.

Chapter 9 proposes and evaluates an algorithm that efficiently finds an approximation and can handle large data sets.

10.3 Conclusions and Contributions on the Meta Level

On a meta level, this dissertation gives an example of how to approach a class of learning problems that is characterized by a certain type of high-dimensional feature space. While this contribution is difficult to evaluate, this dissertation may also provide insight into other tasks with properties similar to text classification. Examples of such tasks could be optical character recognition, natural language understanding tasks, and speech recognition.

10.4 Summary

This dissertation presents a new approach to learning text classifiers from examples. It provides not only learning methods that empirically have state-of-the-art predictive performance, but also a theory that connects the properties of text-classification tasks with the generalization accuracy of the learning methods, as well as algorithms that efficiently implement the methods. While the work presented in this dissertation is driven by the application, the techniques it develops are not limited to text classification, giving it a contribution beyond this particular application. But not only can individual techniques be transferred to other tasks, this dissertation as a whole serves as a case study for how to approach high-dimensional learning tasks.

10.5 Open Question

While this dissertation answers some questions, it also opens new areas for research. What follows is an incomplete list of interesting open questions.

- What are the learning theoretical properties of other learning algorithms for text classification? In particular, it should be possible to analyze other margin-based algorithms like Boosting and Winnow in the same way it was done here for support vector machines.

- What are better learning methods depending on the properties of the task? A promising choice are SVMs with alternative norms. My conjecture is that L_1-margin should work best for tasks with a small set of strong features (e.g. the word "wheat" occurs in all positive examples, but in no negative example). On the opposite side of the spectrum lie L_∞-SVM for extremely dense target concepts.

- How can be amount of training data be further reduced? It was already demonstrated that active learning can reduce the required number of labeled training examples. It might be possible to integrate active learning into the transductive setting for support vector machines.

- Are there lower bounds for the generalization performance of a support vector machine for text classification? In connection with analyzing the theoretical properties of other learning algorithms for text classification, such analysis could identify for which type of task other methods are more appropriate than SVMs.

- Are there special kernels for text classification? While the conventional kernels did not lead to improved performance for text classification, incorporating prior knowledge about text could be beneficial. String kernels [Haussler, 1999][Watkins, 2000] that exploit ordering in the document could be beneficial especially for short documents and for information extraction tasks.

- Can training nonlinear support vector machines be further sped up? This dissertation showed that the largest bottleneck in training nonlinear support vector machines lies in the computation of the kernel matrix. Incorporating information retrieval techniques like an inverted index could provide a speed up for kernels based on dot products. Another promising direction are efficient approximations of the Hessian [Smola and Schölkopf, 2000][Williams and Seeger, 2000].

- How well do humans perform when learning text classifiers merely from examples? In particular, it would be interesting to study how the bias of our learning methods relates to that of humans. Text classification provides a promising ground for such a comparison.

- Is it possible to incorporate prior knowledge about a particular task into the SVM? Often, it is easy to get a natural language description of the classification task. In a relevance feedback setting one might have a query like “documents on machine learning”. The success of relevance feedback demonstrates that this information can be exploited for improved performance. However, machine learning algorithms and theory cannot operationalize such information in a principled way, yet.

- How can we design learning algorithms that directly maximizes precision and recall instead of minimizing error rate?

- How can learning many classes in parallel be sped up? Currently, an individual SVM has to be trained for each class.

- Is it possible to improve prediction performance by considering multiple tasks in parallel? Often, learning tasks are not isolated. Transferring knowledge between tasks was found beneficial in other domains already (see [Caruana et al., 1997]). For example, in text classification it might lead to weighting schemes adapted to a particular collection.

- Do our machine learning algorithms perform well in the real world? The ultimate challenge is making machine learning work beyond our benchmarks and idealized environments. For example, it would be interesting to apply transductive SVMs in relevance feedback and have text classification support service hotlines.

FreeWais Stoplist

The following list of stopwords is used in the experiments reported here.

a
about
above
according
across
actually
adj
after
afterwards
again
against
all
almost
alone
along
already
also
although
always
among
amongst
an
and
another
any
anyhow
anyone
anything
anywhere
are
aren’t
around
as
at
b
be
became
because
become
becomes
becoming
been
before
beforehand
begin
beginning
behind
being
below
beside
besides
between
beyond
billion
both
but
by
c
can
can’t
cannot
caption
co
co.
could
couldn’t
d
did
didn’t
do
does
doesn’t
don’t
down
during
e
each
eg
eight
eighty
either
else
elsewhere
end
ending
enough
etc
even
ever
every
everyone
everything
everywhere
except
f
few
fifty
first
five
for
former
formerly
forty
found
four
from
further
g
h
had
has
hasn’t
have
haven’t
he
he’d
he’ll
he’s
hence
her
here
here’s
hereafter
hereby
herein
hereupon
hers
herself
him
himself
his
how
however
hundred
i
i’d
i’ll
i’m
i’ve
ie
if
in
inc.
indeed
instead
into
is
isn’t
it
it’s
its
itself

j
k
l
last
later
latter
latterly
least
less
let
let’s
like
likely
ltd
m
made
make
makes
many
maybe
me
meantime
meanwhile
might
million
miss
more
moreover
most
mostly
mr
mrs
much
must
my
myself
n
namely
neither
never
nevertheless
next
nine
ninety
no
nobody
none
nonetheless
noone
nor
not
nothing
now
nowhere
o
of
off
often
on
once
one
one’s
only
onto
or
other
others
otherwise
our
ours
ourselves
out
over
overall
own
p
per
perhaps
q
r
rather
recent
recently
s
same
seem
seemed
seeming
seems
seven
seventy
several
she
she’d
she’ll
she’s
should
shouldn’t
since
six
sixty
so
some
somehow
someone
something
sometime
sometimes
somewhere
still
stop
such
t
taking
ten
than
that
that’ll
that’s
that’ve
the
their
them
themselves
then
thence
there
there’d
there’ll
there’re
there’s
there’ve
thereafter
thereby
therefore
therein
thereupon
these
they
they’d
they’ll
they’re
they’ve
thirty
this
those
though
thousand
three
through
throughout
thru
thus
to
together
too
toward
towards
trillion
twenty
two
u
under
unless
unlike
unlikely
until
up
upon
us
used
using
v
very
via
w
was
wasn’t
we
we’d
we’ll
we’re
we’ve
well
were
weren’t
what
what’ll
what’s
what’ve
whatever
when
whence
whenever
where
where’s
whereafter
whereas
whereby
wherein

whereupon
wherever
whether
which
while
whither
who
who’d
who’ll
who’s
whoever
whole
whom
whomever
whose
why
will
with
within
without
won’t
would
wouldn’t
x
y
yes
yet
you
you’d
you’ll
you’re
you’ve
your
yours
yourself
yourselves
z

Bibliography

[Apté and Damerau, 1994] Apté, C. and Damerau, F. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251.

[Araújo et al., 1997] Araújo, M. D., Navarro, G., and Ziviani, N. (1997). Large text searching allowing errors. In Baeza-Yates, R., editor, *Proceedings of the 4th South American Workshop on String Processing*, pages 2–20, Valparaiso, Chile. Carleton University Press.

[Armstrong et al., 1995] Armstrong, R., Freitag, D., Joachims, T., and Mitchell, T. (1995). WebWatcher: A learning apprentice for the World Wide Web. In *Proceedings of the 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, Stanford.

[Armstrong et al., 1998] Armstrong, R., Freitag, D., Joachims, T., and Mitchell, T. (1998). WebWatcher: A learning apprentice for the World Wide Web. In Michalski, R., Bratko, I., and Kubat, M., editors, *Machine Learning and Data Mining*, pages 297–312. Wiley. The file is a copy of Armstrong/etal/95a. Armstrong/etal/98a is a reprint of the 95a document.

[Baeza-Yates and Navarro, 1997] Baeza-Yates, R. and Navarro, G. (1997). Block addressing indices for approximate text retrieval. In Golshani, F. and Makki, K., editors, *Proceedings of the 6th International Conference on Information and Knowledge Management (CIKM-97)*, pages 1–8, New York. ACM Press.

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval.* Addison-Wesley-Longman, Harlow, UK.

[Bailey and Elkan, 1993] Bailey, T. and Elkan, C. (1993). Estimating the accuracy of learned concepts. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 895–900. Morgan Kaufman.

[Baker and McCallum, 1998] Baker, L. D. and McCallum, A. K. (1998). Distributional clustering of words for text classification. In Croft, W. B., Moffat, A., van Rijsbergen, C. J., Wilkinson, R., and Zobel, J., editors, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 96–103, Melbourne, AU. ACM Press, New York, US.

[Balabanovic and Shoham, 1995] Balabanovic, M. and Shoham, Y. (1995). Learning information retrieval agents: Experiments with automated web browsing. In *Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments.* AAAI-Press.

[Basili et al., 1999] Basili, R., Moschitti, A., and Pazienza, M. (1999). A text classifier based on linguistic processing. In Joachims, T., Sahami, M., Ungar, L., and McCallum, A., editors, *IJCAI Workshop on Machine Learning for Information Filtering.*

[Bennett, 1999] Bennett, K. (1999). Combining support vector and mathematical programming methods for classification. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning.* MIT-Press.

[Bennett and Demiriz, 1998] Bennett, K. and Demiriz, A. (1998). Semi-supervised support vector machines. In *Proceedings of Neural Information Processing Systems (NIPS).*

[Berry et al., 1995] Berry, M., Dumais, S., and Shippy, A. (1995). A case study of latent semantic indexing. Technical Report CS-95-271, Computer Science Department, University of Tennessee at Knoxville.

[Blair, 1992] Blair, D. (1992). Information retrieval and the philosophy of language. *The Computer Journal*, 35(3).

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Annual Conference on Computational Learning Theory (COLT-98).*

[Bookstein and Swanson, 1974] Bookstein, A. and Swanson, D. R. (1974). Probabilistic models for automated indexing. *Journal of the American Society for Information Science*, 25(5):312–318.

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A traininig algorithm for optimal margin classifiers. In Haussler, D., editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152.

[Boyan et al., 1996] Boyan, J., Freitag, D., and Joachims, T. (1996). A machine learning architecture for optimizing web search engines. In *AAAI Workshop on Internet Based Information Systems.*

[Brachman and Schmolze, 1985] Brachman, R. and Schmolze, J. (1985). An overview of the kl-one knowledge representation system. *Cognitive Science*, 9(2):171–216.

[Breiman et al., 1984] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and regression trees.* Wadsworth & Brooks, Pacific Grove.

[Buckley et al., 1994] Buckley, C., Salton, G., and Allan, J. (1994). The effect of adding relevance information in a relevance feedback environment. In *International ACM SIGIR Conference*, pages 292–300.

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

[Burges and Crisp, 1999] Burges, C. and Crisp, D. (1999). Uniqueness of the SVM solution. In *Conference on Neural Information Processing Systems (NIPS99).*

[Busemann et al., 2000] Busemann, S., Schmeier, S., and Arens, R. G. (2000). Message classification in the call center. In *6th Conference on Applied Natural Language Processing.*

[Caruana and Freitag, 1994] Caruana, R. and Freitag, D. (1994). Greedy attribute selection. In *International Conference on Machine Learning (ICML).*

[Caruana et al., 1997] Caruana, R., Pratt, L., and Thrun, S. (1997). Multitask learning. *Machine Learning*, 28:41.

[Cataltepe and Magdon-Ismail, 1998] Cataltepe, Z. and Magdon-Ismail, M. (1998). Incorporating test inputs into learning. In Jordan, M. I., Kearns, M. J., and Solla, S. A., editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press.

[Chang et al., 1999] Chang, C.-C., Hsu, C.-W., and Lin, C.-J. (1999). The analysis of decomposition methods for support vector machines. In Saunders, C., editor, *IJCAI99 Workshop on Support Vector Machines*, pages 17–22.

[Chang and Lin, 1999] Chang, C.-C. and Lin, C.-J. (1999). Some analysis on ν-support vector classification. to appear.

[Cheeseman et al., 1988] Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D. (1988). Autoclass: A bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning (ICML)*, pages 54–56.

[Church, 1995] Church, K. W. (1995). One term or two? In Fox, E. A., Ingwersen, P., and Fidel, R., editors, *Proceedings of the 18th Annual International Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 310–318, New York, NY, USA. ACM Press.

[Cohen, 1995] Cohen, W. (1995). Learning to classify english text with ilp methods. In *Advances in Inductive Logic Programming*. IOS Press.

[Cohen, 1996] Cohen, W. (1996). Context-sensitive learning methods for text categorization. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[Cooper, 1991] Cooper, W. (1991). Some inconsistencies and misnomers in probabilistic information retrieval. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 57–61.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. N. (1995). Support–vector networks. *Machine Learning Journal*, 20:273–297.

[Cover and Thomas, 1991] Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. Wiley.

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

[Croft and Lewis, 1990] Croft, W. and Lewis, D. (1990). Term clustering of syntactic phrases. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 385–404.

[Crouch, 1988] Crouch, C. (1988). A cluster-based approach to thesaurus construction. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 309–320.

[Davison and Hall, 1992] Davison, A. and Hall, P. (1992). On the bias and variability of bootstrap and cross-validation estimates of error rate in discriminant problems. *Biometrika*, 79(2):279–284.

[de Buenaga Rodríguez et al., 1997] de Buenaga Rodríguez, M., Gómez-Hidalgo, J. M., and Díaz-Agudo, B. (1997). Using WordNet to complement training information in text categorization. In Milkov, R., Nicolov, N., and Nikolov, N., editors, *Proceedings of RANLP-97, 2nd International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, BL.

[de Kroon et al., 1996] de Kroon, E., Mitchell, T., and Kerckhoffs, E. (1996). Improving learning accuracy in information filtering. In *International Conference on Machine Learning (ICML), Workshop on Machine Learning meets Human Computer Interaction*, pages 41–58.

[de Sa, 1993] de Sa, V. R. (1993). Learning classification with unlabeled data. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Proc. NIPS'93, Neural Information Processing Systems*, pages 112–119, San Francisco, CA. Morgan Kaufmann Publishers.

[Deerwester et al., 1990] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

[D'Espo, 1959] D'Espo, D. (1959). A convex programming procedure. *Naval Research Logistics Quarterly*, 6:33–42.

[Devroye et al., 1996] Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition.* Springer.

[Devroye and Wagner, 1976] Devroye, L. and Wagner, T. (1976). A distribution-free performance bound in error estimation. *IEEE Transactions on Information Theory*, 22:586–587.

[Devroye and Wagner, 1979a] Devroye, L. and Wagner, T. (1979a). Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Transactions on Information Theory*, 25(2):202–207.

[Devroye and Wagner, 1979b] Devroye, L. and Wagner, T. (1979b). Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604.

[Domingos and Pazzani, 1997] Domingos, P. and Pazzani, M. (1997). On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130.

[Drucker et al., 1999] Drucker, H., Wu, D., and Vapnik, V. (1999). Support vector machines for spam categorization. *IEEE Trans. on Neural Networks*, 10(5):1048–1054.

[Dumais, 1994] Dumais, S. (1994). Latent semantic indexing (lsi) and trec-2. Technical Report TM-ARH-023878, Bellcore.

[Dumais et al., 1998] Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of ACM-CIKM98.*

[Efron, 1982] Efron, B. (1982). *The Jackknife, the Bootstrap, and Other Resampling Plans.* SIAM, Philadelphia.

[Efron, 1983] Efron, B. (1983). Estimating the error rate of a prediction rule: Improvements on cross-validation. *Journal of the American Statistical Association*, 78:316–331.

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

[Fagan, 1987] Fagan, J. (1987). *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and non-Syntactic Methods*. PhD thesis, Department of Computer Science, Cornell University.

[Fisher, 1994] Fisher, D. E. (1994). Topic characterization of full length texts using direct and indirect term evidence. Technical Report CSD-94-809, University of California, Berkeley.

[Fletcher, 1987] Fletcher, R. (1987). *Practical Methods of Optimization*. Wiley, 2 edition.

[Foltz, 1990] Foltz, P. (1990). Using latent semantic indexing for information filtering. In *Conference on Office Information Systems*, pages 40–47.

[Foltz and Dumais, 1992] Foltz, P. and Dumais, S. (1992). Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, 35:51–60.

[Forster, 2000] Forster, M. (2000). Key concepts in model selection: Performance and generalization. *Journal of Mathematical Psychology*, 44:205–231.

[Freund and Schapire, 1996] Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann.

[Frieß et al., 1998] Frieß, T.-T., Cristianini, N., and Campbell, C. (1998). The Kernel-Adatron algorithm: a fast and simple learning procedure for Support Vector machines. In *Proc. 15th International Conf. on Machine Learning*, pages 188–196. Morgan Kaufmann, San Francisco, CA.

[Fuhr, 1989] Fuhr, N. (1989). Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems*, 7(3):183–204.

[Fuhr and Buckley, 1991] Fuhr, N. and Buckley, C. (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9(3):223–248.

[Fuhr et al., 1991] Fuhr, N., Hartmann, S., Lustig, G., Schwantner, M., Tzeras, K., and Knorz, G. (1991). Air/x - a rule-based multistage indexing system for large subject fields. In *RIAO*, pages 606–623.

[Fuhr and Knorz, 1984] Fuhr, N. and Knorz, G. (1984). Retrieval test evaluation of a rule based automatic indexing (air/phys). In van Rijsbergen, C., editor, *Research and Development in Information Retrieval: Proceedings of the Third Joint BCS and ACM Symposium*, pages 391–408. Cambridge University Press.

[Fuhr et al., 1994] Fuhr, N., Pfeifer, U., Bremkamp, C., Pollmann, M., and Buckley, C. (1994). Probabilistic learning approaches for indexing and retrieval with the TREC-2 collection. In *The Second Text Retrieval Conference (TREC-2)*. National Institute of Standards and Technology.

[Fung and Mangasarian, 1999] Fung, G. and Mangasarian, O. (1999). Semi-supervised support vector machines for unlabeled data classification. Technical report, Data Mining Institute.

[Fürnkranz et al., 1998] Fürnkranz, J., Mitchell, T., and Riloff, E. (1998). A case study in using linguistic phrases for text categorization on the www. In Sahami, M., Craven, M., Joachims, T., and McCallum, A., editors, *ICML/AAAI Workshop on Learning for Text Categorization.* AAAI Press.

[Gammerman et al., 1998] Gammerman, A., Vapnik, V., and Vowk, V. (1998). Learning by transduction. In *Conference on Uncertainty in Artificial Intelligence*, pages 148–156.

[Gill et al., 1981] Gill, P. E., Murray, W., and Wright, M. H. (1981). *Practical Optimization.* Academic Press.

[Gövert et al., 1999] Gövert, N., Lalmas, M., and Fuhr, N. (1999). A probabilistic description-oriented approach for categorising Web documents. In *Proceedings of CIKM-99, 8th ACM International Conference on Information and Knowledge Management*, pages 475–482, Kansas City, US. ACM Press, New York, US.

[Graepel et al., 2000] Graepel, T., Herbrich, R., and Obermayer, K. (2000). Bayesian transduction. In *Advances in Neural Information System Processing (NIPS99)*, volume 12.

[Grove and Schuurmans, 1998] Grove, A. J. and Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 692–699, Menlo Park. AAAI Press.

[Hammond et al., 1995] Hammond, K., Burke, R., Martin, C., and Lytinen, S. (1995). FAQ finder: A case-based approach to knowledge navigation. In *Proceedings of the Eleventh Conference on Artificial Intelligence for Applications*, pages 80–86, Los Alamitos. IEEE Computer Society Press.

[Haneke, 1999] Haneke, E. (1999). *NewsSIEVE: Ein selbstadaptiver Filter für textuelle Informationen.* PhD thesis, Universität Bonn.

[Harter, 1975a] Harter, S. P. (1975a). A probabilistic approach to automated keyword indexing. Part I: on the distribution of specialty words in a technical literature. *Journal of the American Society for Information Science*, 26(4):197–206.

[Harter, 1975b] Harter, S. P. (1975b). A probabilistic approach to automated keyword indexing. Part II: An algorithm for probabilistic indexing. *Journal of the American Society for Information Science*, 26(5):280–289.

[Haussler, 1999] Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, UC Santa Cruz.

[Hayes and Weinstein, 1990] Hayes, P. and Weinstein, S. (1990). Construe/tis: a system for content-based indexing of a database of news stories. In *Annual Conference on Innovative Applications of AI.*

[Heaps, 1978] Heaps, H. S. (1978). *Information Retrieval: Computational and Theoretical Aspects.* Academic Press, New York.

[Hildreth, 1957] Hildreth, C. (1957). A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4:79–85.

[Hoeffding, 1963] Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30.

[Hsu and Lin, 2000] Hsu, C.-W. and Lin, C.-J. (2000). A simple decomposition method for support vector machines. to appear.

[Jaakkola and Haussler, 1999] Jaakkola, T. and Haussler, D. (1999). Probabilistic kernel regression models. In *Conference on AI and Statistics*.

[Joachims, 1996] Joachims, T. (1996). Einsatz eines intelligenten, lernenden Agenten für das World Wide Web. Diplomarbeit, Fachbereich Informatik, Universität Dortmund.

[Joachims, 1997a] Joachims, T. (1997a). A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of International Conference on Machine Learning (ICML)*.

[Joachims, 1997b] Joachims, T. (1997b). Text categorization with support vector machines: Learning with many relevant features. LS8-Report 23, Universität Dortmund, LS VIII-Report.

[Joachims, 1998a] Joachims, T. (1998a). Making large-scale svm learning practical. LS8-Report 24, Universität Dortmund, LS VIII-Report.

[Joachims, 1998b] Joachims, T. (1998b). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137 – 142, Berlin. Springer.

[Joachims, 1999a] Joachims, T. (1999a). Aktuelles schlagwort: Support vector machines. *Künstliche Intelligenz*, 4.

[Joachims, 1999b] Joachims, T. (1999b). Making Large-Scale SVM Learning Practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169 – 184. MIT Press, Cambridge.

[Joachims, 1999c] Joachims, T. (1999c). Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning (ICML)*, Bled, Slowenien.

[Joachims, 1999d] Joachims, T. (1999d). Wissenserlangung aus grossen datenbanken. In W.Kuckelt and K.Hankeln, editors, *9th Int. Symposium on Intensive Care*, Journal f. Anaesthesie und Intensivbehandlung, Lengerich, Berlin, Riga, Scottsdale (Az.), Wien, Zagreb. Pabst Science Publishers.

[Joachims, 2000] Joachims, T. (2000). Estimating the generalization performance of a SVM efficiently. In *Proceedings of the International Conference on Machine Learning*, San Francisco. Morgan Kaufman.

[Joachims et al., 1997] Joachims, T., Freitag, D., and Mitchell, T. (1997). WebWatcher: a tour guide for the world wide web. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, volume 1, pages 770 – 777. Morgan Kaufmann.

[Joachims et al., 1999] Joachims, T., McCallum, A., Sahami, M., and Craven, M., editors (1999). *Machine Learning for Information Filtering*, IJCAI Workshop. AAAI Press.

[Joachims et al., 1995] Joachims, T., Mitchell, T., Freitag, D., and Armstrong, R. (1995). WebWatcher: Machine learning and hypertext. *Beiträge zum 7. Fachgruppentreffen MASCHINELLES LERNEN der GI-Fachgruppe 1.1.3*, pages 145 – 149. Forschungsbericht Nr. 580 der Universität Dortmund.

[Joachims and Mladenič, 1998] Joachims, T. and Mladenič, D. (1998). Browsing-assistenten, tour guides und adaptive www-server. *Künstliche Intelligenz*, 3(28):23 – 29.

[Junker and Abecker, 1997] Junker, M. and Abecker, A. (1997). Integrating a thesaurus for rule induction in text classification. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Posters, page 338.

[Kaufman, 1999] Kaufman, L. (1999). Solving the quadratic programming problem arising in support vector classification. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 10. MIT-Press.

[Kearns, 1996] Kearns, M. (1996). A bound on the error of cross validation using the approximation and estimation rates, with consequences for the training-test split. In *Advances in Neural Information Processing Systems 8*, pages 183–18. MIT Press.

[Kearns et al., 1997] Kearns, M., Mansour, Y., Ng, A., and Ron, D. (1997). An experimental and theoretical comparison of model selection methods. *Machine Learning*, 27(1):7–50.

[Kearns and Ron, 1997] Kearns, M. and Ron, D. (1997). Algorithmic stability and sanity-check bounds for leave-one-out cross validation. In *Conference on Computational Learning Theory (COLT97)*, pages 152–162.

[Keerthi et al., 1999] Keerthi, S., Shevade, S., Bhattacharyya, C., and Murthy, K. (1999). A fast iterative nearest point algorithm for support vector machine classifier design. Technical Report TR-ISL-99-03, Indian Institute of Science, Bangalore, India.

[Kindermann et al., 2000] Kindermann, J., Diederich, J., Leopold, E., and Paass, G. (2000). Authorship attribution with support vector machines. In *The Learning Workshop at Snowbird*.

[Kivinen et al., 1997] Kivinen, J., Warmuth, M. K., and Auer, P. (1997). The perception algorithm versus winnow: Linear versus logarithmic mistake bounds when few input variables are relevant. *Artificial Intelligence*, 97(1–2):325–343.

[Klinkenberg and Joachims, 2000] Klinkenberg, R. and Joachims, T. (2000). Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, San Francisco. Morgan Kaufmann.

[Kohavi, 1995] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan Kaufman.

[Kreßel, 1999] Kreßel, U. (1999). Pairwise classification and support vector machines. In Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, *Advances in Kernel Methods — Support Vector Learning*, pages 255–268, Cambridge, MA. MIT Press.

[Lachenbruch and Mickey, 1968] Lachenbruch, P. and Mickey, A. (1968). Estimation of error rates in discriminant analysis. *Technometrics*, 10:1–11.

[Lang, 1995] Lang, K. (1995). Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning (ICML).*

[Laskov, 2000] Laskov, P. (2000). An improved decomposition algorithm for regression support vector machines. In Solla, S. A., Leen, T. K., and Müller, K.-R., editors, *Advances in Neural Information Processing (NIPS99)*, volume 12, pages 484–490.

[Lerner and Lawrence, 2000] Lerner, B. and Lawrence, N. D. (2000). A comparison of state-of-the-art classification techniques with application to cytogenetics. *Neural Computing and Applications.* to appear.

[Lewis, 1992a] Lewis, D. (1992a). An evaluation of phrasal and clustered representations on a text categorization task. In *International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[Lewis, 1992b] Lewis, D. (1992b). *Representation and Learning in Information Retrieval.* PhD thesis, Department of Computer and Information Science, University of Massachusetts.

[Lewis, 1992c] Lewis, D. (1992c). *Representation and Learning in Information Retrieval.* PhD thesis, Department of Computer and Information Science, University of Massachusetts.

[Lewis and Gale, 1994] Lewis, D. and Gale, W. (1994). A sequential algorithm for training text classifiers. In *International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[Lewis and Ringuette, 1994] Lewis, D. and Ringuette, M. (1994). A comparison of two learning algorithms for text classification. In *Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93.

[Lewis, 1995] Lewis, D. D. (1995). Evaluating and optmizing autonomous text classification systems. In Fox, E. A., Ingwersen, P., and Fidel, R., editors, *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 246–254, Seattle, US. ACM Press, New York, US.

[Lewis, 1998] Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. In Nédellec, C. and Rouveirol, C., editors, *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, volume 1398 of *LNAI*, pages 4–18, Berlin. Springer.

[Lieberman, 1995] Lieberman, H. (1995). Letizia: An agent that assists Web browsing. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI '95)*, Montreal, Canada. Morgan Kaufmann.

[Liere, 1999] Liere, R. (1999). *Active Learning with Committees: An Approch to Efficient Learning in Text Classification Using Linear Threshold Algorithms.* PhD thesis, Oregon State University.

[Lin, 2000] Lin, C. J. (2000). On the convergence of the decomposition method for support vector machines. Technical report, National Taiwan University, Department of Computer Science and Information Engineering.

[Lunts and Brailovskiy, 1967] Lunts, A. and Brailovskiy, V. (1967). Evaluation of attributes obtained in statistical decision rules. *Engineering Cybernetics*, 3:98–109.

[Lyons, 1968] Lyons, J. (1968). *Introdutions to Theoretical Linguistics.* Cambridge University Press, London.

[Mandelbrot, 1959] Mandelbrot, B. (1959). A note on a class of skew distribution functions: Analysis and critique of a paper by H. A. Simon. *Information and Control*, 2(1):90–99.

[Mangasarian and Musicant, 1999] Mangasarian, O. L. and Musicant, D. R. (1999). Successive overrelaxation for support vector machines. *IEEE-NN*, 10(5):1032.

[Mangasarian and Musicant, 2000] Mangasarian, O. L. and Musicant, D. R. (2000). Active support vector machine classification. Technical Report 00-04, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-04.ps.

[Maron, 1961] Maron, M. E. (1961). Automatic indexing: An experimental inquiry. *Journal of the Association for Computing Machinery*, 8:404–417.

[Masand et al., 1992] Masand, B., Linoff, G., and Waltz, D. (1992). Classifying news stories using memory based reasoning. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–65.

[McCallum and Nigam, 1998] McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. In Sahami, M., Craven, M., Joachims, T., and McCallum, A., editors, *Workshop Notes of the ICML/AAAI-98 Workshop* Learning for Text Categorization, pages 41–48, Menlo Park, CA, USA. AAAI Press.

[Michie et al., 1994] Michie, D., Spiegelhalter, D., and Taylor, C. (1994). *Machine Learning, Neural and Statistical Classification.* Ellis Horwood.

[Miller and Uyar, 1997] Miller, D. and Uyar, S. (1997). A mixture of experts classifier with learning based on both labelled and unlabelled data. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*, pages 571–578, 55 Hayward St., Cambridge, MA, 02142-1399. MIT Press.

[Miller et al., 1958] Miller, G. A., Newman, E. B., and Friedman, E. A. (1958). Length-frequency statistics for written English. *Information and Control*, 1(4):370–389.

[Miller et al., 1990] Miller, G. A., R., B., Fellbaum, C., Gross, D., and Miller, K. J. (1990). Introduction to wordnet: An on-line lexical database. *Journal of Lexicography*, 3(4):234–244.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning.* McGraw Hill, New York.

[Mladenić, 1998] Mladenić, D. (1998). *Machine Learning on Non-Homogeneous, Distributed Text Data.* PhD thesis, University of Ljubljana.

[Moler et al., 2000] Moler, E., Chow, M., and Mian, I. (2000). Analysis of molecular profile data using generative and discriminative methods. *Physiological Genomics.*

[Mood et al., 1974] Mood, A., Graybill, F., and Boes, D. (1974). *Introduction to the Theory of Statistics.* McGraw-Hill, 3 edition.

[Morik et al., 1999] Morik, K., Brockhausen, P., and Joachims, T. (1999). Combining statistical learning with a knowledge-based approach – A case study in intensive care monitoring. In *Proc. 16th Int'l Conf. on Machine Learning (ICML-99)*, Bled, Slowenien.

[Morik et al., 2000] Morik, K., Imhoff, M., Brockhausen, P., Joachims, T., and Gather, U. (2000). Knowledge discovery and knowledge validation in intensive care. *Artificial Intelligence in Medicine.* accepted for publication.

[Moulinier and Ganascia, 1996] Moulinier, I. and Ganascia, J. (1996). Applying an existing machine learning algorithm to text categorization. In Wermter, S., Riloff, E., and Scheler, G., editors, *Connectionist, statistical, and symbolic approaches to learning for natural language processing.* Springer-Verlag.

[Moulinier et al., 1996] Moulinier, I., Raskinis, G., and Ganascia, J. (1996). Text categorization: A symbolic approach. In *Annual Symposium on Document Analysis and Information Retrieval (SDAIR).*

[Neumann and Schmeier, 1999] Neumann, G. and Schmeier, S. (1999). Combining shallow text processing and machine learning in real world applications. In Joachims, T., McCallum, A., Sahami, M., and Ungar, L., editors, *IJCAI99 Workshop on Machine Learning for Information Filtering.*

[Nigam et al., 1998] Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. In *Proceedings of the AAAI-98.*

[Nigam et al., 2000] Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. M. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3).

[Opper and Winther, 2000] Opper, M. and Winther, O. (2000). Gaussian process classification and SVM: Mean field results and leave-one-out estimator. In Bartlett, P., Schölkopf, B., Schuurmans, D., and Smola, A., editors, *Large margin classifiers.* MIT Press, Cambridge, MA.

[Osuna et al., 1996] Osuna, E., Freund, R., and Girosi, F. (1996). Support vector machines: Training and applications. A.I. Memo (in press), MIT A. I. Lab.

[Osuna et al., 1997a] Osuna, E., Freund, R., and Girosi, F. (1997a). An improved training algorithm for support vector machines. In Principe, J., Giles, L., Morgan, N., and Wilson, E., editors, *Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop*, pages 276 – 285, New York. IEEE.

[Osuna et al., 1997b] Osuna, E., Freund, R., and Girosi, F. (1997b). Training support vector machines: An application to face detection. In *Proceedings CVPR'97.*

[Papadimitriou et al., 1998] Papadimitriou, C. H., Raghavan, P., Tamaki, H., and Vempala, S. (1998). Latent semantic indexing: A probabilistic analysis. In ACM, editor, *PODS '98. Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June, 1998, Seattle, Washington*, pages 159–168, New York, NY 10036, USA. ACM Press.

[Pazzani et al., 1996] Pazzani, M., Muramatsu, J., and Billsus, D. (1996). Syskill & webert: Identifying interesting web sites. In *Proc. of the Thirteenth National Conference on Artificial Intelligence (AAAI)*, pages 54–61, Portland.

[Platt, 1998] Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research.

[Platt, 1999a] Platt, J. (1999a). Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 12. MIT-Press.

[Platt, 1999b] Platt, J. (1999b). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Smola, A., Bartlett, P., Scholkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*. MIT Press.

[Platt, 1999c] Platt, J. (1999c). Using sparseness and analytic qp to speed training of support vector machines. In M. S. Kearns, S. A. Solla, D. A. C., editor, *Advances in Neural Information Processing Systems (NIPS98)*, volume 11. MIT Press.

[Platt et al., 2000] Platt, J., Cristianini, N., and Shawe-Taylor, J. (2000). Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems 12*.

[Porter, 1980] Porter, M. (1980). An algorithm for suffix stripping. *Program (Automated Library and Information Systems)*, 14(3):130–137.

[Provost and Fawcett, 1997] Provost, F. and Fawcett, T. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In Heckerman, D., Mannila, H., Pregibon, D., and Uthurusamy, R., editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 43. AAAI Press.

[Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Machine Learning. Morgan Kaufmann, San Mateo, CA.

[Quinlan, 1986] Quinlan, R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

[Quinlan and Cameron-Jones, 1993] Quinlan, R. and Cameron-Jones, R. (1993). Foil: A midterm report. In *European Conference on Machine Learning (ECML)*.

[Raghavan et al., 1989] Raghavan, V., Bollmann, P., and Jung, G. (1989). A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems*, 7(3):205–229.

[Ratsaby and Venkatesh, 1995] Ratsaby, J. and Venkatesh, S. (1995). Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Conference on Computational Learning Theory (COLT)*.

[Riloff and Lehnert, 1994] Riloff, E. and Lehnert, W. (1994). Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems*, 12(3):296–333.

[Robertson, 1977] Robertson, S. (1977). The probability ranking principle in ir. *Journal of Documentation*, 33(4):294–304.

[Rocchio, 1971] Rocchio, J. (1971). Relevance feedback in information retrieval. In Salton, G., editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, USA.

[Rogers and Wagner, 1978] Rogers, W. and Wagner, T. (1978). A finite sample distribution-free performance bound for local discrimination rules. *Annals of Statistics*, 6:506–514.

[Sahami, 1998] Sahami, M. (1998). *Using Machine Learning to Improve Information Access*. PhD thesis, Stanford University.

[Sahami et al., 1998] Sahami, M., Craven, M., Joachims, T., and McCallum, A., editors (1998). *Learning for Text Categorization*, number WS-98-05 in ICML/AAAI Workshop. AAAI Press.

[Salton, 1971] Salton, G., editor (1971). *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Englewood Cliffs, NJ.

[Salton, 1991] Salton, G. (1991). Developments in automatic text retrieval. *Science*, 253:974–979.

[Salton and Buckley, 1988] Salton, G. and Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523.

[Saunders et al., 1999] Saunders, C., Gammerman, A., and Vovk, V. (1999). Transduction with confidence and credibility. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 722–726, San Francisco. Morgan Kaufman.

[Schapire et al., 1997] Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. (1997). Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann.

[Schapire and Singer, 2000] Schapire, R. E. and Singer, Y. (2000). BoosTexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

[Schapire et al., 1998] Schapire, R. E., Singer, Y., and Singhal, A. (1998). Boosting and Rocchio applied to text filtering. In Croft, W. B., Moffat, A., van Rijsbergen, C. J., Wilkinson, R., and Zobel, J., editors, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 215–223, Melbourne, AU. ACM Press, New York, US.

[Scheffer and Joachims, 1999] Scheffer, T. and Joachims, T. (1999). Expected error analysis for model selection. In *International Conference on Machine Learning (ICML)*, Bled, Slowenien.

[Schűtze et al., 1995] Schűtze, H., Hull, D., and Pedersen, J. (1995). A comparison of classifiers and document representations for the routing problem. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[Schölkopf, 1997] Schölkopf, B. (1997). *Support Vector Learning*. R. Oldenbourg Verlag, Munich.

[Schölkopf et al., 1999] Schölkopf, B., Smola, A., and Williamson, R. (1999). Shrinking the tube: a new support vector regression algorithm. In M. S. Kearns, S. A. Solla, D. A. C., editor, *Advances in Neural Information Processing (NIPS98)*, volume 11.

[Schölkopf et al., 2000] Schölkopf, B., Smola, A., Williamson, R., and Bartlett, P. (2000). New support vector algorithms. *Neural Computation*, 12:1083–1121.

[Scott and Matwin, 1998] Scott, S. and Matwin, S. (1998). Text classification using WordNet hypernyms. In Harabagiu, S., editor, *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pages 38–44. Association for Computational Linguistics, Somerset, New Jersey.

[Shahshahani and Landgrebe, 1994] Shahshahani, B. and Landgrebe, D. (1994). The effect of unlabeled samples in reducing the small sample size problem and mitigating the hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, 32(5):1087–1095.

[Shao and Tu, 1995] Shao, J. and Tu, D. (1995). *The Jackknife and Bootstrap.* Springer, New York.

[Shawe-Taylor et al., 1996] Shawe-Taylor, J., Bartlett, P., Williamson, R., and Anthony, M. (1996). Structural risk minimization over data-dependent hierarchies. Technical Report NC-TR-96-053, NeuroCOLT.

[Smola, 1998] Smola, A. (1998). *Learning with Kernels.* PhD thesis, Technische Universität Berlin.

[Smola and Schölkopf, 2000] Smola, A. and Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *International Conference on Machine Learning*, pages 911–918.

[Spark-Jones, 1973] Spark-Jones, K. (1973). Collection properties influencing automatic term classification performance. *Information Storage and Retrieval*, 9:499–513.

[Stone, 1974] Stone, M. (1974). Cross-validatory choice and assesment of statistical predictions. *Journal of the Royal Statistical Society Series B*, 36:111–147.

[Stone, 1977] Stone, M. (1977). Asymptotics for and against cross-validation. *Biometrika*, 64(1):29–35.

[Syed et al., 1999] Syed, N. A., Liu, H., and Sung, K. K. (1999). A study of support vectors on model independent example selection. In Chaudhuri, S. and Madigan, D., editors, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 272–276, N.Y. ACM Press.

[Taira and Haruno, 1999] Taira, H. and Haruno, M. (1999). Feature selection in svm text categorization. In *Conference of the American Association for Artificial Intelligence (AAAI)*, pages 480–486.

[Toussaint and Donaldson, 1970] Toussaint, G. and Donaldson, R. (1970). Algorithms for recognizing contour-traced handprinted characters. *IEEE Transactions on Computers*, 19:541–546.

[Tzeras and Hartmann, 1993] Tzeras, K. and Hartmann, S. (1993). Automatic indexing based on bayesian inference networks. In *Proceedings of the ACM SIGIR*, pages 22–34.

[van Rijsbergen, 1977] van Rijsbergen, C. (1977). A theoretical basis for the use of cooccurrence data in information retrieval. *Journal of Documentation*, 33(2):106–119.

[van Rijsbergen, 1979] van Rijsbergen, C. (1979). *Information Retrieval.* Butterworths, London, 2. edition.

[van Rijsbergen et al., 1981] van Rijsbergen, C., Harper, D., and Porter, M. (1981). The selection of good search terms. *Information Processing and Management*, 17:77–91.

[Vanderbei, 1994] Vanderbei, R. (1994). Loqo: An interior point code for quadratic programming. Technical report, Princeton University.

[Vapnik, 1982] Vapnik, V. (1982). *Estimation of Dependencies Based on Empirical Data.* Springer.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory.* Wiley, Chichester, GB.

[Vapnik and Sterin, 1977] Vapnik, V. and Sterin, A. (1977). On structural risk minimization or overall risk in a problem of pattern recognition. *Automation and Remote Control*, 10(3):1495–1503.

[Vapnik, 1995] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory.* Springer, New York.

[Vovk et al., 1999] Vovk, V., Gammerman, A., and Saunders, C. (1999). Machine-learning applications of algorithmic randomness. In *Proc. 16th International Conf. on Machine Learning*, pages 444–453. Morgan Kaufmann, San Francisco, CA.

[Wahba, 1999] Wahba, G. (1999). Support vector machines, reproducing kernel hilbert spaces, and randomized gacv. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 6, pages 69–88. MIT-Press.

[Wapnik and Tscherwonenkis, 1979] Wapnik, W. and Tscherwonenkis, A. (1979). *Theorie der Zeichenerkennung.* Akademie Verlag, Berlin.

[Watkins, 2000] Watkins, C. (2000). Dynamic alignment kernels. In Smola, A., Bartlett, P., Schölkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 39–50. MIT Press, Cambridge, MA.

[Weiss and Indurkhya, 1993] Weiss, S. and Indurkhya, N. (1993). Optimized rule induction. *IEEE Expert*, 8(6):61–69.

[Weiss et al., 1999] Weiss, S. M., Apté, C., Damerau, F. J., Johnson, D. E., Oles, F. J., Goetz, T., and Hampp, T. (1999). Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):63–69.

[Werner, 1984] Werner, J. (1984). *Optimization - Theory and Applications.* Vieweg.

[Weston and Watkins, 1998] Weston, J. and Watkins, C. (1998). Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway University of London.

[Whorf, 1959] Whorf, B. L. (1959). *Language, Thought, and Reality.* Whiley.

[Wiener et al., 1995] Wiener, E., Pedersen, J., and Weigend, A. (1995). A neural network approach to topic spotting. In *Annual Symposium on Document Analysis and Information Retrieval (SDAIR).*

[Williams and Seeger, 2000] Williams, C. and Seeger, M. (2000). The effekt of the input density distribution on kernel-based classifiers. In *International Conference on Machine Learning*, pages 1159–1166.

[Wismer and Chattergy, 1978] Wismer, D. and Chattergy, R. (1978). *Introduction to Nonlinear Optimization.* North-Holland, New York.

[Wittgenstein, 1967] Wittgenstein, L. (1967). *Philosophical Investigations.* Blackwell, Oxford, 2 edition.

[Wolfe, 1972] Wolfe, P. (1972). On the convergence of gradient methods under constraint. *IBM Journal on Research and Development*, 16:407–411.

[Wu et al., 1999] Wu, D., Bennett, K. P., Cristianini, N., and Shawe-Taylor, J. (1999). Large margin trees for induction and transduction. In *Proc. 16th International Conf. on Machine Learning*, pages 474–483. Morgan Kaufmann, San Francisco, CA.

[Yang, 1995] Yang, Y. (1995). Noise reduction in a statistical approach to text categorization. In *Proceedings of the ACM SIGIR on Research and Development in Information Retrieval.*

[Yang, 1997] Yang, Y. (1997). An evaluation of statistical approaches to text categorization. Technical report, Carnegie Mellon University.

[Yang, 1999] Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69–90.

[Yang and Chute, 1993] Yang, Y. and Chute, C. (1993). Words or concepts: the features of indexing units and their optimal use in information retrieval. In *Annual Symposium on Computer Applications in Medical Care (SCAMC)*, pages 685–689.

[Yang and Liu, 1999] Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In *International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[Yang and Pedersen, 1997] Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In Fisher, D. H., editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US. Morgan Kaufmann Publishers, San Francisco, US.

[Zipf, 1949] Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology.* Addison-Wesley Press Inc., Cambridge 42, MA, USA.

[Zoutendijk, 1970] Zoutendijk, G. (1970). *Mathematical Programming Methods.* North-Holland.