# Estimating the Generalization Performance of a SVM Efficiently

Thorsten Joachims
Universität Dortmund
LS Informatik VIII
44221 Dortmund, Germany
thorsten@ls8.informatik.uni-dortmund.de

**Abstract**

This paper proposes and analyzes an approach to estimating the generalization performance of a support vector machine (SVM) for text classification. Without any computation intensive resampling, the new estimators are computationally 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 estimators delevoped here address the special performance measures needed for text classification. While they can be used to estimate error rate, one can also estimate the recall, the precision, and the $F_1$. A theoretical analysis and experiments on three text classification collections show that the new method can effectively estimate the performance of SVM text classifiers in a very efficient way.

## 1 Introduction

Predicting the generalization performance of a learner is one of the central goals of learning theory. From a practical perspective, a learning theory should provide accurate and efficient methods for predicting how well a learner can handle the 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 following presents an approach to answering these questions in the context of text classification with SVMs [Joachims, 1998][Dumais et al., 1998]. The aim is to develop very operational performance estimators that are of actual use when applying SVMs. This requires that the estimators are both effective and computationally efficient. While the results presented in the following are general enough (or can easily be generalized) to 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 cover 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 paper is structured as follows. After a short description of the text classification problem, methods for estimating the generalization error of a learner are reviewed in section 3. While some of these (e.g. uniform convergence bounds) are powerful theoretical methods, they are of little use in practical applications. Others (e.g. cross-validation, bootstrap) give good predictions, but are computationally inefficient. Section 5 decribes

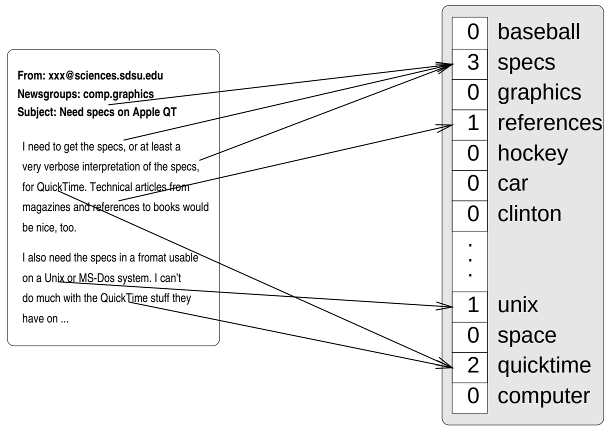


Figure 1: Representing text as a feature vector.

new estimators that 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 can be computed efficiently. After their theoretical justification, the estimators are experimentally tested on three text classification tasks in section 6. The experiments show that they very accurately reflect the actual behavior of SVMs on text classification tasks.

# 2 Text Classification

The goal of text classification is the automatic assignment of documents to a fixed number of semantic categories. Each document can be in multiple, exactly one, or no category at all. Using machine learning, the objective is to learn classifiers from examples which assign categories automatically. To facilitate effective and efficient learning, each category is treated as a separate binary classification problem. Each such problem answers the question of whether or not a document should be assigned to a particular category. This leads to the following type of supervised learning problem. Given is an i.i.d. training sample $S_n$ of size $n$

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n) \tag{1}$$

with $\vec{x}_i$ representing the document content and $y_i \in \{-1, +1\}$ indicating the class. The learner $\mathcal{L}$ aims to find a decision rule $h_{\mathcal{L}}(\vec{x})$ based on $S_n$ that classifies new documents as accurately as possible.

## 2.1 Representation

Documents, which typically are strings of characters, have to be transformed into a representation suitable for the learning algorithm and the classification task. Information Retrieval research suggests that words work well as representation units and that for many tasks their ordering can be ignored without losing too much information. This leads to an attribute-value representation of text. Each distinct word $w_j$ corresponds to a feature with $TF(w_j, d_i)$, the number of times word $w_j$ occurs in the document $d_i$, as its value. Figure 1 shows an example feature vector for a particular document.

Refining this basic representation, it has been shown that scaling the dimensions of the feature vector with their *inverse document frequency* $IDF(w_j)$ [Salton and Buckley, 1988] leads to an improved performance. $IDF(w_j)$ can be calculated from the document

| | label +1 | label -1 | |
|---|---|---|---|
| predict +1 | $\Pr(h(\vec{x})=1,y=1)$ | $\Pr(h(\vec{x})=1,y=-1)$ | $\Pr(h(\vec{x})=1)$ |
| predict -1 | $\Pr(h(\vec{x})=-1,y=1)$ | $\Pr(h(\vec{x})=-1,y=-1)$ | $\Pr(h(\vec{x})=-1)$ |
| | $\Pr(y=1)$ | $\Pr(y=-1)$ | |

Figure 2: Contingency table.

frequency $DF(w_j)$, which is the number of documents the word $w_j$ occurs in.

$$IDF(w_j) = log\left(\frac{n}{DF(w_j)}\right) \tag{2}$$

Here, $n$ is the total number of documents. Intuitively, the inverse document frequency of a word is low if it occurs in many documents and is highest if the word occurs in only one. To abstract from different document lengths, each document feature vector $\vec{x}_i$ is normalized to unit length.

### 2.2 Performance Measures

Unlike for other applications of machine learning, error rate alone is not necessarily a good performance measure in text classification. Instead, scores based on precision and recall are of widespread use. Based on the contingency table in figure 2, I define the recall $Rec(h)$ of a decision rule $h$ as the probability that a document with label $y=1$ is classified correctly.

$$Rec(h) = \frac{\Pr(h(\vec{x})=1,y=1)}{\Pr(h(\vec{x})=1,y=1)+\Pr(h(\vec{x})=-1,y=1)} \tag{3}$$

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

$$Prec(h) = \frac{\Pr(h(\vec{x})=1,y=1)}{\Pr(h(\vec{x})=1,y=1)+\Pr(h(\vec{x})=1,y=-1)} \tag{4}$$

Between high precision and high recall exists a trade-off. To get a single performance measure, the geometric mean of precision and recall is commonly used. It is called the $F1$ measure and can be written as follows.

$$F1(h) = \frac{2\ \Pr(h(\vec{x})=1,y=1)}{2\ \Pr(h(\vec{x})=1,y=1)+\Pr(h(\vec{x})=-1,y=1)+\Pr(h(\vec{x})=1,y=-1)} \tag{5}$$

## 3 Generic Performance Estimators

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

$$Err^n(h_{\mathcal{L}}) = \int L(h_{\mathcal{L}}(\vec{x}),y)d\Pr(\vec{x},y) = \Pr(h_{\mathcal{L}}(\vec{x}) \neq y|S_n) \tag{6}$$

of a learner $\mathcal{L}$ based on a sample $S_n$ of size $n$, with $L$ being the 0/1-loss function. In particular, these are uniform convergence bounds for the training error (section 3.1), as well as Hold-Out Testing (section 3.2), Bootstrapping (section 3.3), and Cross-Validation (section 3.4).

### 3.1 Training Error

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

$$Err^n_{emp}(h_\mathcal{L}) = \frac{1}{n}\sum_{i=1}^{n} L(h_\mathcal{L}(\vec{x}_i), y_i) \tag{7}$$

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 strong optimistic bias.

$$\mathcal{E}(Err^n_{emp}(h_\mathcal{L})) << \mathcal{E}(Err^n(h_\mathcal{L})) \tag{8}$$

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 that 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 upper bounds the difference. The bounds depend 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 they are 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:

$$\Pr(|Err^n(h_\mathcal{L}) - Err^n_{emp}(h_\mathcal{L})| > \epsilon) \leq 6\left(\frac{e\,n}{d_\mathcal{H}}\right)^{d_\mathcal{H}} exp\left(-\frac{\epsilon^2(n-1)}{4}\right) \tag{9}$$

Using this bound can help quantify, by how much $Err^n_{emp}(h_\mathcal{L})$ underestimates the true error. This leads to an upper bound on the error rate as it is desired in practice. Solving (9) for $Err^n(h_\mathcal{L})$ gives such an upper bound. With probability $1 - \eta$

$$Err^n(h_\mathcal{L}) \leq Err^n_{emp}(h_\mathcal{L}) + 2\sqrt{\frac{d_\mathcal{H}(ln\frac{2\,n}{d_\mathcal{H}} + 1) - ln\frac{\eta}{4}}{n}} \tag{10}$$

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

Let's 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 (9) into:

$$\mathcal{E}(|Err^n(h_\mathcal{L}) - Err^n_{emp}(h_\mathcal{L})|^2) \leq O\left(\frac{d_\mathcal{H}\,ln(\frac{n}{d_\mathcal{H}})}{n}\right) \tag{11}$$

### 3.2 Hold-Out Testing

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

$$Err^{l,k}_{ho}(h_\mathcal{L}) = \frac{1}{k}\sum_{(\vec{x}_i, y_i)\in S^{val}_k} L(h_\mathcal{L}(\vec{x}_i), y_i) \tag{12}$$

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 only with $l$ 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.

$$\mathcal{E}(Err_{ho}^{l,k}(h_\mathcal{L})) > \mathcal{E}(Err^n(h_\mathcal{L})) \tag{13}$$

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.

$$\mathcal{E}(|Err^l(h_\mathcal{L}) - Err_{ho}^{l,k}(h_\mathcal{L})|^2) \leq \frac{1}{4\,k} \tag{14}$$

Equation (13) and (14) exhibit a trade-off in selecting $l$ and $k$. The larger $l$, the smaller the bias. At the same time, the variance increases with $k = n - l$ decreasing. 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's finally also look at worst case bounds for the deviation. Using Hoeffding bounds [Hoeffding, 1963] it holds that

$$\Pr(|Err^l(h_\mathcal{L}) - Err_{ho}^{l,k}(h_\mathcal{L})| > \epsilon) \leq 2\,exp(-2\,k\,\epsilon^2) \tag{15}$$

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.

### 3.3 Bootstrap and Jackknife

The methods based on boostrap [Efron, 1983][Efron, 1982][Shao and Tu, 1995] or jackknife [Efron, 1982] statistics try to estimate the bias of the training error $Err_{emp}^n(h_\mathcal{L})$. All bootstrap methods make use of bootstrap samples $S_m^b = ((\vec{x}_1^b, y_1^b), \cdots, (\vec{x}_m^b, y_m^b))$, 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 $h_\mathcal{L}^i$. For the $k$-th bootstrap sample the estimate of the bias is

$$bias_k^n = \sum_{i=1}^{n} \left( \frac{1}{n} - \frac{1}{n} \sum_{j=1}^{m} L(\vec{x}_j^b, \vec{x}_i) \right) L(h_\mathcal{L}^i(\vec{x}_i), y_i) \tag{16}$$

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

$$bias^n = \sum_{k=1}^{B} bias_k^n \tag{17}$$

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

$$R_b^n(h_\mathcal{L}) = Err_{emp}^n(h_\mathcal{L}) - bias^n \tag{18}$$

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 $R_b^n(h_{\mathcal{L}})$. 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.

### 3.4 Cross-Validation and Leave-One-Out

The most popular method for estimating the generalization error of a decision rule is cross-validation [Lunts and Brailovskiy, 1967][Stone, 1974][Lachenbruch and Mickey, 1968] (delete-d method, rotation estimate). While there are several versions of the cross-validation estimator, most theoretical results concern the leave-one-out (loo) 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 $S^{\setminus 1} = ((\vec{x}_1, y_1), \cdots, (\vec{x}_n, y_n))$ is used for training, leading to a classification rule $h_{\mathcal{L}}^{\setminus 1}$. 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 devided by $n$ is the loo-estimate of the generalization error.

$$Err_{loo}^n(h_{\mathcal{L}}) = \frac{1}{n} \sum_{i=1}^{n} L(h_{\mathcal{L}}^{\setminus i}(\vec{x}_i), y_i) \tag{19}$$

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

**Theorem 1 ( Bias of Leave-One-Out Estimator)** *[Lunts and Brailovskiy, 1967]. The leave-one-out estimator is almost unbiased; that is*

$$\mathcal{E}(R_{loo}^n(h_{\mathcal{L}})) = \mathcal{E}(R^{n-1}(h_{\mathcal{L}})) \tag{20}$$

*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:*

$$\begin{aligned}
\mathcal{E}(R_{loo}^n(h_{\mathcal{L}})) &= \int \frac{1}{n} \sum_{i=1}^{n} L(h_{\mathcal{L}}^{\setminus i}(\vec{x}_i), y_i) dP(Z_1)...dP(Z_n) && (21)\\
&= \frac{1}{n} \sum_{i=1}^{n} \int L(h_{\mathcal{L}}^{\setminus i}(\vec{x}_i), y_i) dP(Z_1)...dP(Z_n) && (22)\\
&= \frac{1}{n} \sum_{i=1}^{n} \int \left[ \int L(h_{\mathcal{L}}^{\setminus i}(\vec{x}_i), y_i) dP(Z_i) \right] dP(Z_1)...dP(Z_{i-1}) dP(Z_{i+1}) && (23)\\
&\quad ...dP(Z_n) && (24)\\
&= \frac{1}{n} \sum_{i=1}^{n} \int R^{n-1}(h_{\mathcal{L}}) dP(Z_1)...dP(Z_{i-1}) dP(Z_{i+1})...dP(Z_n) && (25)\\
&= \frac{1}{n} n \int R^{n-1}(h_{\mathcal{L}}) dP(Z_1)...dP(Z_{n-1}) && (26)\\
&= \mathcal{E}(R^{n-1}(h_{\mathcal{L}})) && (27)
\end{aligned}$$

∎

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 neglectably small.

The variability of the loo-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 2 (Variability of the LOO-Estimator)** *[Rogers and Wagner, 1978][Devroye and Wagner, 1976]. It holds that*

$$\mathcal{E}(|Err^n_{loo}(h_{\mathcal{L}}) - Err^n(h_{\mathcal{L}})|^2) \leq \frac{1}{n} + 6\ \Pr(h^n_{\mathcal{L}}(\vec{x}) \neq h^{n-1}_{\mathcal{L}}(\vec{x})) \tag{28}$$

$\Pr(h^n_{\mathcal{L}}(\vec{x}) \neq h^{n-1}_{\mathcal{L}}(\vec{x}))$ is the probability that a decision rule $h^n_{\mathcal{L}}(\vec{x})$ (trained using all $n$ examples) will disagree on a randomly drawn example with a decision rule $h^{n-1}_{\mathcal{L}}(\vec{x}))$ (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 2 can be used to upper bound the variability of the loo-estimator for the special case of local decision rules [Devroye and Wagner, 1979b][Devroye and Wagner, 1979a]. Somewhat surprising results about the asymptotic behavior of the loo-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 loo-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 5.1.

The computational demands of the loo-estimator are high. The learner is invoked $n$ times on training sets of $n-1$ examples. This is prohibitively expensive for all but very 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 decision rule is tested on the remaining fold. The average performance is the $k$-fold cross validation estimate $Err^n_{kcv}(h_{\mathcal{L}})$ (delete-$(\frac{n}{k})$ estimate, rotation estimate). Note that $k$-fold cross validation has a larger bias than leave-one-out.

$$\mathcal{E}(Err^n_{kcv}(h_{\mathcal{L}})) = \mathcal{E}(R^{n-\frac{n}{k}}(h_{\mathcal{L}})) \tag{29}$$

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].

# 4 Support Vector Machines

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. In their basic form, SVMs learn linear decision rules

$$h(\vec{x}) = sign\{\vec{w}\cdot\vec{x}+b\} = \begin{cases} +1, & \text{if } \vec{w}\cdot\vec{x}+b>0 \\ -1, & \text{else} \end{cases} \tag{30}$$

described by a weight vector $\vec{w}$ and a threshold $b$. The idea of structural risk minimization is to find a hypothesis $h$ for which one can guarantee the lowest probability of error. For SVMs, Vapnik shows that this goal can be translated into finding the hyperplane with maximum soft-margin[1]. Computing this hyperplane is equivalent to solving the following optimization problem:

**Optimization Problem 1 (Soft-Margin SVM (primal))**

$$\textit{minimize:} \quad V(\vec{w}, \vec{\xi}) = \vec{w} \cdot \vec{w} + C \sum_{i=1}^{n} \xi_i \tag{31}$$

$$\textit{subject to:} \quad \forall i \in [1..n] : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i \tag{32}$$

$$\forall_{i=1}^{n} : \xi_i > 0 \tag{33}$$

In this optimization problem, the Euclidean length $||\vec{w}||$ of the weight vector is inversely proportional to the soft-margin of the decision rule. The constraints (32) require that all training examples are classified correctly up to some slack $\xi_i$. If a training example lies on the "wrong" side of the hyperplane, the corresponding $\xi_i$ is greater or equal to 1. Therefor $\sum_{i=1}^{n} \xi_i$ is an upper bound on the number of training errors. The factor $C$ in (31) is a parameter that allows trading-off training error vs. model complexity.

For computational reasons it is useful to solve the Wolfe dual [Fletcher, 1987] of optimation problem 1 instead of solving optimation problem 1 directly [Vapnik, 1998].

**Optimization Problem 2 (Soft-Margin SVM (dual))**

$$\textit{minimize:} \quad W(\vec{\alpha}) = -\sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j) \tag{34}$$

$$\textit{subject to:} \quad \sum_{i=1}^{n} y_i \alpha_i = 0 \tag{35}$$

$$\forall i \in [1..n] : 0 \leq \alpha_i \leq C \tag{36}$$

In this paper, $SVM^{Light}$ [Joachims, 1999] is used for computing the solution of this optimization problem[2]. All training examples with $\alpha_i > 0$ at the solution 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 2 the decision rule can be computed as

$$\vec{w} \cdot \vec{x} = \left( \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i \right) \cdot \vec{x} = \sum_{i=1}^{n} \alpha_i y_i (\vec{x}_i \cdot \vec{x}) \quad \text{and} \quad b = y_{usv} - \vec{w} \cdot \vec{x}_{usv} \tag{37}$$

The support vector $(\vec{x}_{usv}, y_{usv})$ for calculating $b$ has be be an unbounded support vector. While it is highly unlikely in practice that one gets a solution of optimization problem 2 with only bounded support vectors, it is theoretically possible (see [Burges and Crisp, 1999][Rifkin et al., 1999] for a thorough 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*.

For both solving optimization problem 2 as well as applying the learned decision rule, it is sufficient to be able to calculate inner products between attribute vectors. Exploiting this property, Boser et al. [Boser et al., 1992] introduced the use of kernels $K(\vec{x}_1, \vec{x}_2)$ for learning non-linear decision rules. Such kernels calculate an inner-product in some feature space and replace the inner-product in the formulas above.

[1] See [Cortes and Vapnik, 1995] for an introduction to SVMs.

[2] $SVM^{Light}$ is available at http://www-ai.informatik.uni-dortmund.de/svm_light

## 5 Efficient Performance Estimators for SVMs

While the estimation methods in the previous section are applicable to arbitrary learning algorithms, this section presents special estimators for Support Vector Machines. The estimators proposed in the following are based on the leave-one-out method, but require and 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 2 and the vector $\vec{\xi}$ from the solution of the primal SVM training problem 1. Due to this dependence, they will be called $\xi\alpha$-estimators in the following.

To get useful tools for text classification I will propose and explore $\xi\alpha$-estimators not only for the error rate (section 5.1), but also for the recall (section 5.2), the precision (section 5.3), and the $F1$-measure (section 5.4).

### 5.1 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 1 ($\xi\alpha$-Estimator of the Error Rate)** *For stable soft-margin SVMs, the $\xi\alpha$-estimator of the error rate is*

$$Err^n_{\xi\alpha}(h_\mathcal{L}) = \frac{d}{n} \quad \textit{with} \quad d = |\{i : (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \tag{38}$$

*with $\rho$ equals 2. $\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 2 and 1 on the training set $S_n$. $R^2_\Delta$ is an upper bound on $\mathcal{K}(\vec{x},\vec{x}) - \mathcal{K}(\vec{x},\vec{x}')$ for all $\vec{x},\vec{x}'$.*

The definition introduces the parameter $\rho$. While the theoretical results that are derived below assume $\rho = 2$, we will see that $\rho = 1$ is a good choice for text classification[3]. The key quantity in definition 1 is $d$. $d$ counts the number of training examples for which the inequality $(\rho\alpha_i R^2 + \xi_i) \geq 1$ 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 $(\rho\alpha_i R^2 + \xi_i) \geq 1$ 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 $S_n^{\setminus i}$, then example $(\vec{x}_i, y_i)$ *must* fulfill the inequality $(\rho\alpha_i R^2 + \xi_i) \geq 1$ 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 1 (Bound on Leave-One-Out Error of Stable Soft-Margin SVMs)** *The number of leave-one-out errors $\sum_{i=1}^n L(h_\mathcal{L}^{\setminus i}(\vec{x}_i), y_i)$ of stable soft-margin SVMs on a training set $S_n$ is bounded by*

$$\sum_{i=1}^{n} L(h_\mathcal{L}^{\setminus i}(\vec{x}_i), y_i) \leq |\{i : (2\,\alpha_i R^2 + \xi_i) \geq 1\}| \tag{39}$$

*$\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 2 and 1 on the training set $S_n$. $R^2$ is an upper bound on $\mathcal{K}(\vec{x},\vec{x})$ and $\mathcal{K}(\vec{x},\vec{x}') \geq 0$.*

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

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

$$W_t(\vec{\alpha}^t) = max_{\vec{0} \le \vec{\alpha}^t \le \vec{C}} 1^T \vec{\alpha}^t - \frac{1}{2} \vec{\alpha}^{tT} H^t \vec{\alpha}^t \wedge \vec{y}^{tT} \vec{\alpha}^t = 0 \tag{40}$$

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

$$y_t \left[ \sum_{i \neq t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b^t \right] > 0 \tag{41}$$

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

$$W(\vec{\alpha}) = max_{\vec{0} \le \vec{\alpha} \le \vec{C}} 1^T \vec{\alpha} - \frac{1}{2} \vec{\alpha}^T H \vec{\alpha} \wedge \vec{y}^T \vec{\alpha} = 0 \tag{42}$$

*that involves all n training examples. Three cases can occur based on the optimal value of* $\alpha_t$*:*

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* $y_t \sum_{i \neq t} y_i \alpha_i \mathcal{K}(\vec{x}_t, \vec{x}_i) \ge 1$ *and so (41) 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* $W(.)$.

$$\beta_i = \begin{cases} \alpha_i^t & if \quad \alpha_i^t = 0 \vee \alpha_i^t = C \\ \alpha_i^t - y_i y_t \nu_i & if \quad i \in SV^t \\ \alpha_i & if \quad i = t \end{cases} \tag{43}$$

$\nu_i$ *has to fulfill the following constraints. Let's* $SV^t$ *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* $\nu_i$ *are chosen to be non-negative and so that* $\sum_{i \in SV^t} \nu_i = \alpha_t$ *and* $0 \le \beta_i \le C$. *Finding such* $\nu_i$ *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* $\nu_i$ *it follows that* $\vec{y}^T \vec{\beta} = 0$ *and* $0 \le \beta_i \le C$. *So* $\vec{\beta}$ *is a feasible point of* $W(.)$. *After a series of transformations,* $W(\vec{\beta})$ *can be written as*

$$W(\vec{\beta}) = W_t(\vec{\alpha}^t) - \frac{1}{2} \alpha_t^2 \mathcal{K}(\vec{x}_t, \vec{x}_t) + \alpha_t \tag{44}$$

$$-\alpha_t y_t \sum_{i \in SV^t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) - y_t \left[ \sum_{i \in SV^t} \nu_i \left( y_i - \sum_{j \in SV^t} \alpha_j^t y_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \right) \right] \tag{45}$$

$$-\frac{1}{2} \sum_{i \in SV^t} \sum_{j \in SV^t} \nu_i \nu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \alpha_t \sum_{i \in SV^t} \nu_i \mathcal{K}(\vec{x}_i, \vec{x}_t) \tag{46}$$

*For support vectors not at the upper bound the expression in the round brackets (line 45) equals the threshold* $b^t$ *of the decision rule (compare equation (37)). Exploiting also that* $\sum_{i \in SV^t} \nu_i = \alpha_t$ *by construction, it is possible to write:*

$$\alpha_t y_t \left[ \sum_{i \in SV^t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b^t \right] = -W(\vec{\beta}) + W_t(\vec{\alpha}^t) - \frac{1}{2} \alpha_t^2 \mathcal{K}(\vec{x}_t, \vec{x}_t) + \alpha_t \tag{47}$$

$$-\frac{1}{2} \sum_{i \in SV^t} \sum_{j \in SV^t} \nu_i \nu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \tag{48}$$

$$+\alpha_t \sum_{i \in SV^t} \nu_i \mathcal{K}(\vec{x}_i, \vec{x}_t) \tag{49}$$

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

$$\gamma_i = \begin{cases} \alpha_i & if \quad \alpha_i = 0 \vee \alpha_i = C \\ \alpha_i + y_i y_t \mu_i & if \quad i \in SV \setminus \{t\} \end{cases} \tag{50}$$

*$SV$ 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. $\mu_i$ 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 $\mu_i$ 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*

$$\begin{aligned} W^t(\vec{\gamma}) &= W(\vec{\alpha}) + \frac{1}{2}\alpha_t^2 \mathcal{K}(\vec{x}_t, \vec{x}_t) - \alpha_t && (51) \\ &+ \alpha_t y_t \sum_{i \in SV \setminus \{t\}} \alpha_i y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) && (52) \\ &+ y_t \left[ \sum_{i \in SV \setminus \{t\}} \mu_i \left( y_i - \sum_{j \in SV \setminus \{t\}} \alpha_j y_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \right) \right] && (53) \\ &- \frac{1}{2} \sum_{i \in SV \setminus \{t\}} \sum_{j \in SV \setminus \{t\}} \mu_i \mu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \alpha_t \sum_{i \in SV \setminus \{t\}} \mu_i \mathcal{K}(\vec{x}_i, \vec{x}_t) && (54) \end{aligned}$$

*Now, the expression in the round brackets equals the threshold $b$ of the decision rule based on all examples. Substituting and rearraging leads to the equation*

$$\begin{aligned} -W(\vec{\alpha}) &= -W^t(\vec{\gamma}) + \frac{1}{2}\alpha_t^2 \mathcal{K}(\vec{x}_t, \vec{x}_t) - \alpha_t && (55) \\ &+ \alpha_t y_t \left[ \sum_{i \in SV \setminus \{t\}} \alpha_i y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b \right] && (56) \\ &- \frac{1}{2} \sum_{i \in SV \setminus \{t\}} \sum_{j \in SV \setminus \{t\}} \mu_i \mu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \alpha_t \sum_{i \in SV \setminus \{t\}} \mu_i \mathcal{K}(\vec{x}_i, \vec{x}_t) && (57) \end{aligned}$$

*It is now possible to substitute $W(\vec{\alpha})$ for $W(\vec{\beta})$ in equation (47). Since $W(\vec{\alpha})$ is larger than $W(\vec{\beta})$ by definition, this results in the inequality*

$$\begin{aligned} \alpha_t y_t \left[ \sum_{i \in SV^t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b^t \right] &\geq W_t(\vec{\alpha}^t) - W^t(\vec{\gamma}) && (58) \\ &+ \alpha_t y_t \left[ \sum_{i \in SV \setminus \{t\}} \alpha_i y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b \right] && (59) \\ &- \frac{1}{2} \sum_{i \in SV \setminus \{t\}} \sum_{j \in SV \setminus \{t\}} \mu_i \mu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) && (60) \\ &+ \alpha_t \sum_{i \in SV \setminus \{t\}} \mu_i \mathcal{K}(\vec{x}_i, \vec{x}_t) && (61) \\ &- \frac{1}{2} \sum_{i \in SV^t} \sum_{j \in SV^t} \nu_i \nu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) && (62) \\ &+ \alpha_t \sum_{i \in SV^t} \nu_i \mathcal{K}(\vec{x}_i, \vec{x}_t) && (63) \end{aligned}$$

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

$$\alpha_t y_t \left[ \sum_{i \in SV^t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b^t \right] \geq \alpha_t y_t \left[ \sum_{i \in SV \setminus \{t\}} \alpha_i y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b \right] \tag{64}$$

$$
\begin{aligned}
&-\frac{1}{2}\sum_{i\in SV\setminus\{t\}}\sum_{j\in SV\setminus\{t\}}\mu_i\mu_j\mathcal{K}(\vec{x}_i,\vec{x}_j) && (65)\\
&+\alpha_t\sum_{i\in SV\setminus\{t\}}\mu_i\mathcal{K}(\vec{x}_i,\vec{x}_t) && (66)\\
&-\frac{1}{2}\sum_{i\in SV^t}\sum_{j\in SV^t}\nu_i\nu_j\mathcal{K}(\vec{x}_i,\vec{x}_j) && (67)\\
&+\alpha_t\sum_{i\in SV^t}\nu_i\mathcal{K}(\vec{x}_i,\vec{x}_t) && (68)
\end{aligned}
$$

*The term $\alpha_t \sum_{i\in SV^t}\nu_i\mathcal{K}(\vec{x}_i,\vec{x}_t)$ is non-negative, since all $\nu_i$ 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 or equal to $R^2$. Using these inequalities and dividing by $\alpha_t$, it is possible to write*

$$
y_t\left[\sum_{i\in SV^t}\alpha_i^t y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b^t\right] \geq y_t\left[\sum_{i\in SV\setminus\{t\}}\alpha_i y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b\right]-\alpha_t R^2 \qquad (69)
$$

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

$$
y_t\left[\sum_{i\in SV\setminus\{t\}}\alpha_i y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b\right]-\alpha_t R^2\leq 0 \qquad (70)
$$

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

$$
\begin{aligned}
1 &\leq \alpha_t\mathcal{K}(\vec{x}_t,\vec{x}_t)+\alpha_t R^2 && (71)\\
\Rightarrow 1 &\leq 2\alpha_t R^2 && (72)
\end{aligned}
$$

*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*

$$
y_t\left[\sum_{i\in SV\setminus\{t\}}\alpha_i y_i\mathcal{K}(\vec{x}_t,\vec{x}_i)+b\right]-\alpha_t R^2\leq 0 \qquad (73)
$$

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

$$
\begin{aligned}
1-\xi_t &\leq \alpha_t\mathcal{K}(\vec{x}_t,\vec{x}_t)+\alpha_t R^2 && (74)\\
\Rightarrow 1 &\leq 2\alpha_t R^2+\xi_t && (75)
\end{aligned}
$$

*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 $\vec{\alpha}$ for unbiased hyperplanes goes back to Vapnik (cf. [Vapnik, 1998], pages 418-421). Unlike the work presented here, Vapnik's result is limited to the case where the training data is separable and it is used to derive bounds on the expected error - not estimators. Jaakkola and Haussler [Jaakkola and Haussler, 1999] present a generalized bound for inseparable data that is similar to that of lemma 1. Similarly, an approximation to the leave-one-out error of SVMs was recently proposed in [Wahba, 1999]. Nevertheless, like Vapnik's bound both are restricted to unbiased hyperplanes and do not apply to regular SVMs.

While lemma 1 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 2 (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 semidefinite 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 of 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*

$$\begin{aligned} W(\vec{\alpha}) &= -\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\mathcal{K}(\vec{x}_i, \vec{x}_j) + c) && (76) \\ &= -\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \frac{c}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j && (77) \\ &= -\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \frac{c}{2}\sum_{i=1}^{n} y_i \alpha_i \sum_{j=1}^{n} y_j \alpha_j && (78) \\ &= -\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \mathcal{K}(\vec{x}_i, \vec{x}_j) && (79) \end{aligned}$$

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

$$\begin{aligned} \vec{w}\cdot\vec{x} &= \sum_{i=1}^{n} \alpha_i y_i (\mathcal{K}(\vec{x}_i, \vec{x}_j) + c) && (80) \\ &= \sum_{i=1}^{n} \alpha_i y_i \mathcal{K}(\vec{x}_i, \vec{x}_j) + c\sum_{i=1}^{n}\alpha_i y_i && (81) \\ &= \sum_{i=1}^{n} \alpha_i y_i \mathcal{K}(\vec{x}_i, \vec{x}_j) && (82) \end{aligned}$$

∎

The previous two lemmas and theorem 1 complete the tools needed to characterize the bias of the $\xi\alpha$-estimator of the error rate.

**Theorem 3 (Bias of $\xi\alpha$-Estimator of the Error Rate)** *The $\xi\alpha$-estimator of the error rate is pessimistically biased in the following sense*

$$\mathcal{E}(Err_{\xi\alpha}^{n}(h_{\mathcal{L}})) \geq \mathcal{E}(R^{n-1}(h_{\mathcal{L}})) \tag{83}$$

**Proof** *Theorem 1 shows that the leave-one-out estimator $Err_{loo}^{n}(h_{\mathcal{L}})$ 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 1 using lemma 2. The lemma establishes that $Err_{\xi\alpha}^{n}(h_{\mathcal{L}}) \geq Err_{loo}^{n}(h_{\mathcal{L}})$ with $R^2 = c + max\mathcal{K}(\vec{x}, \vec{x})$ and therefore $\mathcal{E}(Err_{\xi\alpha}^{n}(h_{\mathcal{L}})) \geq \mathcal{E}(Err^{n-1}(h_{\mathcal{L}}))$.* ∎

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 2 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 decision rule with minimum training error, Kearns and Ron bound the variability based on the VC-dimension of the hypothesis space.

**Theorem 4 (Bound on the Variability of $Err^n_{loo}(h_{\mathcal{L}})$)** *[Kearns and Ron, 1997]. 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 $1-\nu$,*

$$|Err^n_{loo}(h_{\mathcal{L}}) - Err^n(h_{\mathcal{L}})| \leq \frac{8\sqrt{\frac{(d+1)(ln(9\,n/d)+2)}{n}}}{\nu} \tag{84}$$

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 lose to be of practical importance. Therefor, the variability of the $\xi\alpha$-estimator will be assessed empirically in section 6.

## 5.2 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 2 ($\xi\alpha$-Estimator of the Recall)** *For stable soft-margin SVMs, the $\xi\alpha$- estimator of the recall is*

$$Rec^n_{\xi\alpha}(h_{\mathcal{L}}) = 1 - \frac{d_+}{n_+} \quad \textit{with} \quad d_+ = |\{i : y_i = 1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \tag{85}$$

$$n_+ = |\{i : y_i = 1\}| \tag{86}$$

*with $\rho$ equals 2. $\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 2 and 1 on the training set $S_n$. $R^2_\Delta$ is an upper bound on $\mathcal{K}(\vec{x},\vec{x}) - \mathcal{K}(\vec{x},\vec{x}')$ for all $\vec{x},\vec{x}'$.*

$d_+$ 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 shows that $d_+$ is an upper bound on the number $loo_+$ of positive examples that produce a leave-one-out error.

**Lemma 3 (Bound on $loo_+$ for Stable Soft-Margin SVMs)** *The number $loo_+$ of leave-one-out errors on positive examples for stable soft-margin SVMs on the training set $S_n$, is bounded by*

$$loo_+ \leq |\{i : y_i = 1 \wedge (2\,\alpha_i R^2 + \xi_i) \geq 1\}| \tag{87}$$

*$\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 2 and 1 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 1 is easily specialized to this case.* ∎

$loo_+$ 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*

$$Err^n_{+loo}(h_\mathcal{L}) = \frac{loo_+}{n} \tag{88}$$

*gives an almost unbiased estimate of the false positive rate $R^{n-1}_+(h_\mathcal{L})$ in the following sense.*

$$\mathcal{E}(Err^n_{+loo}(h_\mathcal{L})) = \mathcal{E}(R^{n-1}_+(h_\mathcal{L})) \tag{89}$$

**Proof** *For the loss function*

$$L_+(h(\vec{x}), y) = \begin{cases} 1 & y = 1 \wedge h(x) = -1 \\ 0 & else \end{cases} \tag{90}$$

*the risks correspond to the false positive rates. Since theorem 1 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 decision 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. 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 3 (Micro-Expected Value of a Function)** *The micro-expected value $\mathcal{E}_{micro}(f(X))$ of a function $f(x)$ is defined as*

$$\mathcal{E}_{micro}(f(X)) = f(\mathcal{E}(X)) \tag{91}$$

*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

$$\mathcal{E}_{micro}(Rec(h)) = \frac{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1))}{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1)) + \mathcal{E}(\Pr(h(\vec{x}) = -1, y = 1))} \tag{92}$$

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

$$\mathcal{E}_{micro}(Rec^n_{\xi\alpha}(h_\mathcal{L})) = 1 - \frac{\mathcal{E}(d_+)}{\mathcal{E}(n_+)} \tag{93}$$

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

$$\mathcal{E}_{micro}(Rec^n_{\xi\alpha}(h_\mathcal{L})) \leq \mathcal{E}_{micro}(Rec^{n-1}(h_\mathcal{L})) \tag{94}$$

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

$$\mathcal{E}_{micro}(Rec(H)) = \frac{\mathcal{E}(\Pr(h(\vec{x})=1, y=1))}{\mathcal{E}(\Pr(h(\vec{x})=1, y=1)) + \mathcal{E}(\Pr(h(\vec{x})=-1, y=1))} \quad (95)$$

$$= \frac{\mathcal{E}(\Pr(h(\vec{x})=1, y=1))}{\mathcal{E}(\Pr(h(\vec{x})=1, y=1) + \Pr(h(\vec{x})=-1, y=1))} \quad (96)$$

$$= \frac{\mathcal{E}(\Pr(h(\vec{x})=1, y=1))}{\mathcal{E}(\Pr(y=1))} \quad (97)$$

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

$$\mathcal{E}(\Pr(h(\vec{x})=1, y=1)) = \mathcal{E}(\Pr(y=1) - \Pr(h(\vec{x})=-1, y=1)) \quad (98)$$

$$= \mathcal{E}(\Pr(y=1)) - \mathcal{E}(\Pr(h(\vec{x})=-1, y=1)) \quad (99)$$

$$= \Pr(y=1) - \mathcal{E}(\Pr(h(\vec{x})=-1, y=1)) \quad (100)$$

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

$$\mathcal{E}_{micro}(Rec^{n-1}(h_{\mathcal{L}})) = \frac{\mathcal{E}(\Pr(h(\vec{x})=1, y=1))}{\mathcal{E}(\Pr(y=1))} \quad (101)$$

$$= \frac{\Pr(y=1) - \mathcal{E}(\Pr(h(\vec{x})=-1, y=1))}{\Pr(y=1)} \quad (102)$$

$$= 1 - \frac{\mathcal{E}(\Pr(h(\vec{x})=-1, y=1))}{\Pr(y=1)} \quad (103)$$

$$\geq 1 - \frac{\frac{\mathcal{E}(d_+)}{n}}{\Pr(y=1)} \quad (104)$$

$$= 1 - \frac{\mathcal{E}(d_+)}{\mathcal{E}(n_+)} \quad (105)$$

$$= \mathcal{E}_{micro}(Rec^n_{\xi\alpha}(h_{\mathcal{L}})) \quad (106)$$

*The inequality holds since* $d_+$ *is an upper bound on* $loo_+$ *(shown in lemma 3, possibly after invariant transformation of the kernel function according to lemma 2), and* $\mathcal{E}(loo_+) = \mathcal{E}(\Pr(h(\vec{x}) = -1, y = 1))$ *(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 6.

## 5.3 Precision

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

**Definition 4 ($\xi\alpha$-Estimator of the Precision)** *For stable soft-margin SVMs, the* $\xi\alpha$*-estimator of the precision is*

$$Prec^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{n_+ - d_+}{n_+ - d_+ + d_-} \quad \textit{with} \quad d_+ = |\{i : y_i = 1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \quad (107)$$

$$d_- = |\{i : y_i = -1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\} \quad (108)$$

$$n_+ = |\{i : y_i = 1\}| \quad (109)$$

*with* $\rho$ *equals 2.* $\vec{\alpha}$ *and* $\vec{\xi}$ *are the solution of optimization problems 2 and 1 on the training set* $S_n$*.* $R^2_\Delta$ *is an upper bound on* $\mathcal{K}(\vec{x}, \vec{x}) - \mathcal{K}(\vec{x}, \vec{x}')$ *for all* $\vec{x}, \vec{x}'$*.*

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

**Lemma 4 (Bound on $loo_-$ for Stable Soft-Margin SVMs)** *The number $loo_-$ of leave-one-out errors on negative examples for stable soft-margin SVM on the training sample $S_n$, is bounded by*

$$loo_- \leq |\{i : y_i = -1 \wedge (\rho\, \alpha_i R^2 + \xi_i) \geq 1\}| \tag{110}$$

*$\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 2 and 1 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 1 is easily specialized to this case.* ∎

Like for 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*

$$Err^n_{-loo}(h_\mathcal{L}) = \frac{loo_-}{n} \tag{111}$$

*gives an almost unbiased estimate of the false positive rate $R^{n-1}_-(h_\mathcal{L})$ in the following sense.*

$$\mathcal{E}(Err^n_{-loo}(h_\mathcal{L})) = \mathcal{E}(R^{n-1}_-(h_\mathcal{L})) \tag{112}$$

**Proof** *For the loss function*

$$L_-(h(\vec{x}), y) = \begin{cases} 1 & y = -1 \wedge h(x) = 1 \\ 0 & else \end{cases} \tag{113}$$

*the risks correspond to the false negative rates. Since theorem 1 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

$$\mathcal{E}_{micro}(Rec(h)) = \frac{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1))}{\mathcal{E}(\Pr(h(\vec{x}) = 1, y = 1)) + \mathcal{E}(\Pr(h(\vec{x}) = 1, y = -1))} \tag{114}$$

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

$$\mathcal{E}_{micro}(Prec^n_{\xi\alpha}(h_\mathcal{L})) = \frac{\mathcal{E}(n_+) - \mathcal{E}(d_+)}{\mathcal{E}(n_+) - \mathcal{E}(d_+) + \mathcal{E}(d_-)} \tag{115}$$

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

**Theorem 6 (Bias of the $\xi\alpha$-Estimator of the Precision)** *The $\xi\alpha$-estimator of the precision $Prec^n_{\xi\alpha}(h_\mathcal{L})$ is pessimistically biased in the following sense:*

$$\mathcal{E}_{micro}(Prec^n_{\xi\alpha}(h_\mathcal{L})) \leq \mathcal{E}_{micro}(Prec^{n-1}(h_\mathcal{L})) \tag{116}$$

**Proof** *Recall from the proof of theorem 5 that*

$$\mathcal{E}(\Pr(h(\vec{x})=1,y=1)) = \Pr(y=1)-\mathcal{E}(\Pr(h(\vec{x})=-1,y=1)) \quad (117)$$

$$\geq \Pr(y=1)-\frac{\mathcal{E}(d_+)}{n} \quad (118)$$

*The following chain of transformations proves the theorem:*

$$\mathcal{E}_{micro}(Prec^{n-1}(h_{\mathcal{L}})) = \frac{\mathcal{E}(\Pr(h(\vec{x})=1,y=1))}{\mathcal{E}(\Pr(h(\vec{x})=1,y=1))+\mathcal{E}(\Pr(h(\vec{x})=1,y=-1))} \quad (119)$$

$$\geq \frac{\Pr(y=1)-\frac{\mathcal{E}(d_+)}{n}}{\Pr(y=1)-\frac{\mathcal{E}(d_+)}{n}+\mathcal{E}(\Pr(h(\vec{x})=1,y=-1))} \quad (120)$$

$$\geq \frac{\Pr(y=1)-\frac{\mathcal{E}(d_+)}{n}}{\Pr(y=1)-\frac{\mathcal{E}(d_+)}{n}+\frac{\mathcal{E}(d_-)}{n}} \quad (121)$$

$$= \frac{\mathcal{E}(n_+)-\mathcal{E}(d_+)}{\mathcal{E}(n_+)-\mathcal{E}(d_+)+\mathcal{E}(d_-)} \quad (122)$$

$$= \mathcal{E}_{micro}(Prec^n_{\xi\alpha}(h_{\mathcal{L}})) \quad (123)$$

*The first inequality holds since $d_+$ is an upper bound on $loo_+$ (shown in lemma 3, possibly after invariant transformation of the kernel function according to lemma 2), and $\mathcal{E}(loo_+)=\mathcal{E}(\Pr(h(\vec{x})=-1,y=1))$ (shown in lemma 1). The second inequality holds since $d_-$ is an upper bound on $loo_-$ (shown in lemma 4), and $\mathcal{E}(loo_-)=\mathcal{E}(\Pr(h(\vec{x})=1,y=-1))$ (shown in lemma 2).* ∎

### 5.4 $F1$-Measure

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

**Definition 5 ($\xi\alpha$-Estimator of the $F1$-Measure)** *For stable soft-margin SVMs, the $\xi\alpha$-estimator of the $F1$-measure is*

$$F1^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{2\,n_+-2\,d_+}{2\,n_+-d_++d_-} \quad \text{with} \quad d_+ = |\{i : y_i=1 \wedge (\rho\alpha_i R^2_\Delta+\xi_i)\geq 1\}| \quad (124)$$

$$d_- = |\{i : y_i=-1 \wedge (\rho\alpha_i R^2_\Delta+\xi_i)\geq 1\} \quad (125)$$

$$n_+ = |\{i : y_i=1\}| \quad (126)$$

*with $\rho$ equals 2. $\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems 2 and 1 on the training set $S_n$. $R^2_\Delta$ is an upper bound on $\mathcal{K}(\vec{x},\vec{x})-\mathcal{K}(\vec{x},\vec{x}')$ for all $\vec{x},\vec{x}'$.*

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

**Theorem 7 (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 pessimistically biased in the following sense:*

$$\mathcal{E}_{micro}(F1^n_{\xi\alpha}(h_{\mathcal{L}})) \leq \mathcal{E}_{micro}(F1^{n-1}(h_{\mathcal{L}})) \quad (127)$$

**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 3, 4, and 2.*

$$\mathcal{E}_{micro}(F1^{n-1}(h_{\mathcal{L}})) = 2\,\mathcal{E}(\Pr(h(\vec{x})=1,y=1))/\,(2\,\mathcal{E}(\Pr(h(\vec{x})=1,y=1))+ \quad (128)$$

$$\mathcal{E}(\Pr(h(\vec{x})=-1,y=1))+\mathcal{E}(\Pr(h(\vec{x})=1,y=-1))) \quad (129)$$

$$\geq \quad 2\,(\Pr(y=1) - \frac{\mathcal{E}(d_+)}{n})/\Big(2\,(\Pr(y=1) - \frac{\mathcal{E}(d_+)}{n}) \tag{130}$$

$$+\mathcal{E}(\Pr(h(\vec{x}) = -1, y = 1)) + \mathcal{E}(\Pr(h(\vec{x}) = 1, y = -1))) \tag{131}$$

$$\geq \quad \frac{2\ \Pr(y=1) - 2\ \frac{\mathcal{E}(d_+)}{n}}{2\ \Pr(y=1) - \frac{\mathcal{E}(d_+)}{n} + \frac{\mathcal{E}(d_-)}{n}} \tag{132}$$

$$= \quad \frac{2\,\mathcal{E}(n_+) - 2\,\mathcal{E}(d_+)}{2\,\mathcal{E}(n_+) - \mathcal{E}(d_+) + \mathcal{E}(d_-)} \tag{133}$$

$$= \quad \mathcal{E}_{micro}(F1^n_{\xi\alpha}(h_{\mathcal{L}})) \tag{134}$$

∎

# 6 Experiments

The following experiments explore how well the $\xi\alpha$-estimators work in practice. The evaluation is done on the three text classification tasks Reuters, WebKB, and Ohsumed. The results for Reuters are discussed in detail. The findings are validated on the other two collections.

The Reuters-21578 dataset[4] was collected from the Reuters newswire in 1987. The "ModApte" subset is used, leading to a corpus of 12,902 documents. Of the 135 potential topic categories only the most frequent 10 are used, while keeping all documents.

The WebKB collection[5] consists 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 third test collection is taken from the Ohsumed corpus[6] compiled by William Hersh. From the 50,216 documents in 1991 which have abstracts, the first 20,000 are used in the following experiments. The task is to assign documents to one or multiple categories of the 5 most frequent MeSH "diseases" categories. A document belongs to a category if it is indexed with at least one indexing term from that category.

Two values for the parameter $\rho$ are evaluated in the following, namely $\rho = 2$ and $\rho = 1$. The setting $\rho = 2$ is a direct consequence of lemma 1, 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

$$\frac{1}{2}\sum_{i\in SV\setminus\{t\}}\ \sum_{j\in SV\setminus\{t\}} \mu_i\mu_j\mathcal{K}(\vec{x}_i,\vec{x}_j) \quad \text{and} \quad \frac{1}{2}\sum_{i\in SV^t}\ \sum_{j\in SV^t} \nu_i\nu_j\mathcal{K}(\vec{x}_i,\vec{x}_j) \tag{135}$$

in the proof of lemma 1. 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 all support vectors have identical feature vectors $\vec{x}_i$. For text classification problems the opposite is true. Most support vectors 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 both expressions in (135) 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.

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 to be able to compare variance estimates. The $\xi\alpha$-estimators

[4] Available at http://www.research.att.com/~lewis/ reuters21578.html

[5] Available at http://www.cs.cmu.edu/afs/cs/project/ theo-20/www/data

[6] Available at ftp://medir.ohsu.edu/pub/ohsumed

| $\rho = 1$ | $Err_{\xi\alpha}(h_\mathcal{L})$ | $Err(h_\mathcal{L})$ | $Rec_{\xi\alpha}(h_\mathcal{L})$ | $Rec(h_\mathcal{L})$ | $Prec_{\xi\alpha}(h_\mathcal{L})$ | $Prec(h_\mathcal{L})$ | $F1_{\xi\alpha}(h_\mathcal{L})$ | $F1(h_\mathcal{L})$ |
|---|---|---|---|---|---|---|---|---|
| earn | 2.68 ± 0.08 | 1.87 ± 0.15 | 92.3 ± 0.3 | 94.8 ± 0.5 | 98.8 ± 0.1 | 99.1 ± 0.1 | 95.4 ± 0.1 | 96.9 ± 0.2 |
| acq | 3.54 ± 0.16 | 2.53 ± 0.20 | 88.1 ± 0.8 | 92.5 ± 0.5 | 92.1 ± 0.4 | 93.6 ± 0.7 | 90.0 ± 0.5 | 93.1 ± 0.4 |
| money-fx | 2.67 ± 0.12 | 1.92 ± 0.14 | 64.9 ± 2.0 | 73.6 ± 2.4 | 83.4 ± 1.4 | 89.8 ± 1.9 | 73.0 ± 1.4 | 80.9 ± 1.4 |
| grain | 1.23 ± 0.09 | 0.82 ± 0.11 | 74.4 ± 1.7 | 82.7 ± 2.1 | 98.2 ± 0.8 | 98.5 ± 0.5 | 84.7 ± 1.2 | 89.9 ± 1.2 |
| crude | 1.58 ± 0.08 | 1.30 ± 0.10 | 72.5 ± 1.7 | 76.6 ± 2.6 | 90.9 ± 1.0 | 92.7 ± 1.2 | 80.6 ± 1.0 | 83.9 ± 1.3 |
| trade | 1.99 ± 0.11 | 1.42 ± 0.10 | 60.4 ± 2.1 | 70.0 ± 2.4 | 83.5 ± 1.3 | 89.0 ± 1.5 | 70.0 ± 1.7 | 78.3 ± 1.6 |
| interest | 2.20 ± 0.18 | 1.67 ± 0.20 | 54.0 ± 5.2 | 63.8 ± 4.3 | 80.2 ± 2.9 | 87.2 ± 2.8 | 64.5 ± 4.4 | 73.6 ± 3.4 |
| ship | 1.23 ± 0.10 | 0.96 ± 0.13 | 47.5 ± 5.6 | 61.2 ± 3.9 | 93.3 ± 1.9 | 93.8 ± 2.5 | 62.7 ± 5.0 | 74.0 ± 2.9 |
| wheat | 0.91 ± 0.07 | 0.65 ± 0.08 | 62.8 ± 1.8 | 71.1 ± 2.3 | 95.0 ± 1.5 | 97.8 ± 1.0 | 75.6 ± 1.2 | 82.3 ± 1.4 |
| corn | 0.90 ± 0.05 | 0.71 ± 0.06 | 52.4 ± 4.1 | 61.8 ± 1.7 | 97.3 ± 1.8 | 99.1 ± 0.6 | 68.1 ± 3.6 | 76.1 ± 1.3 |
| $\rho = 2$ | $Err_{\xi\alpha}(h_\mathcal{L})$ | $Err(h_\mathcal{L})$ | $Rec_{\xi\alpha}(h_\mathcal{L})$ | $Rec(h_\mathcal{L})$ | $Prec_{\xi\alpha}(h_\mathcal{L})$ | $Prec(h_\mathcal{L})$ | $F1_{\xi\alpha}(h_\mathcal{L})$ | $F1(h_\mathcal{L})$ |
| earn | 6.79 ± 0.21 | 1.87 ± 0.15 | 84.4 ± 0.4 | 94.8 ± 0.5 | 92.7 ± 0.5 | 99.1 ± 0.1 | 88.3 ± 0.4 | 96.9 ± 0.2 |
| acq | 12.99 ± 0.28 | 2.53 ± 0.20 | 59.3 ± 1.4 | 92.5 ± 0.5 | 66.0 ± 1.2 | 93.6 ± 0.7 | 62.5 ± 1.2 | 93.1 ± 0.4 |
| money-fx | 5.38 ± 0.22 | 1.92 ± 0.14 | 38.4 ± 2.0 | 73.6 ± 2.4 | 52.1 ± 1.8 | 89.8 ± 1.9 | 44.2 ± 1.8 | 80.9 ± 1.4 |
| grain | 3.20 ± 0.19 | 0.82 ± 0.11 | 48.1 ± 2.0 | 82.7 ± 2.1 | 72.8 ± 1.8 | 98.5 ± 0.5 | 57.9 ± 2.0 | 89.9 ± 1.2 |
| crude | 3.69 ± 0.20 | 1.30 ± 0.10 | 45.3 ± 2.1 | 76.6 ± 2.6 | 62.8 ± 2.4 | 92.7 ± 1.2 | 52.6 ± 2.1 | 83.9 ± 1.3 |
| trade | 3.80 ± 0.15 | 1.42 ± 0.10 | 34.7 ± 1.8 | 70.0 ± 2.4 | 51.2 ± 2.2 | 89.0 ± 1.5 | 41.4 ± 1.9 | 78.3 ± 1.6 |
| interest | 4.19 ± 0.26 | 1.67 ± 0.20 | 27.5 ± 4.2 | 63.8 ± 4.3 | 40.8 ± 5.0 | 87.2 ± 2.8 | 32.8 ± 4.6 | 73.6 ± 3.4 |
| ship | 2.21 ± 0.08 | 0.96 ± 0.13 | 18.2 ± 2.6 | 61.2 ± 3.9 | 49.4 ± 5.2 | 93.8 ± 2.5 | 26.6 ± 3.5 | 74.0 ± 2.9 |
| wheat | 1.79 ± 0.14 | 0.65 ± 0.08 | 43.2 ± 1.8 | 71.1 ± 2.3 | 65.8 ± 2.7 | 97.8 ± 1.0 | 52.2 ± 1.9 | 82.3 ± 1.4 |
| corn | 1.72 ± 0.12 | 0.71 ± 0.06 | 28.1 ± 2.1 | 61.8 ± 1.7 | 57.1 ± 3.2 | 99.1 ± 0.6 | 37.6 ± 2.2 | 76.1 ± 1.3 |

Table 1: Table comparing average $\xi\alpha$-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 upper half shows the estimates for $\rho = 1$, the lower half for $\rho = 2$. The "true" values are estimated from a holdout 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.

are applied to the SVM trained on the training set. The test set is used to get a holdout 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 $C$ is chosen since it was selected by the $\xi\alpha$-estimates in model selection experiments [Joachims, 2000]. 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] as described in [Joachims, 1998]. The resulting document vectors are normalized to unit length. This implies $R_\Delta = 1$.

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

Figure 3 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 desired. For the 100 experiments (10 splits for 10 classes) the estimate of the error was lower than the error measured on the holdout 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 holdout estimates, this indicates a small variance of the $\xi\alpha$-estimate, in particular, since the holdout estimate is subject to variance, too. This is also supported by table 1, which gives additional details on the results. The top half of the table contains the $\xi\alpha$-estimates with $\rho = 1$, the lower half with $\rho = 2$. For $\rho = 2$ the $\xi\alpha$-estimates are substantially more biased than for $\rho = 1$. 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 holdout estimates, especially for $\rho = 1$. This shows that in terms of variance the $\xi\alpha$-estimates are as good as holdout testing without requiring an additional test set of the same size as the training data.

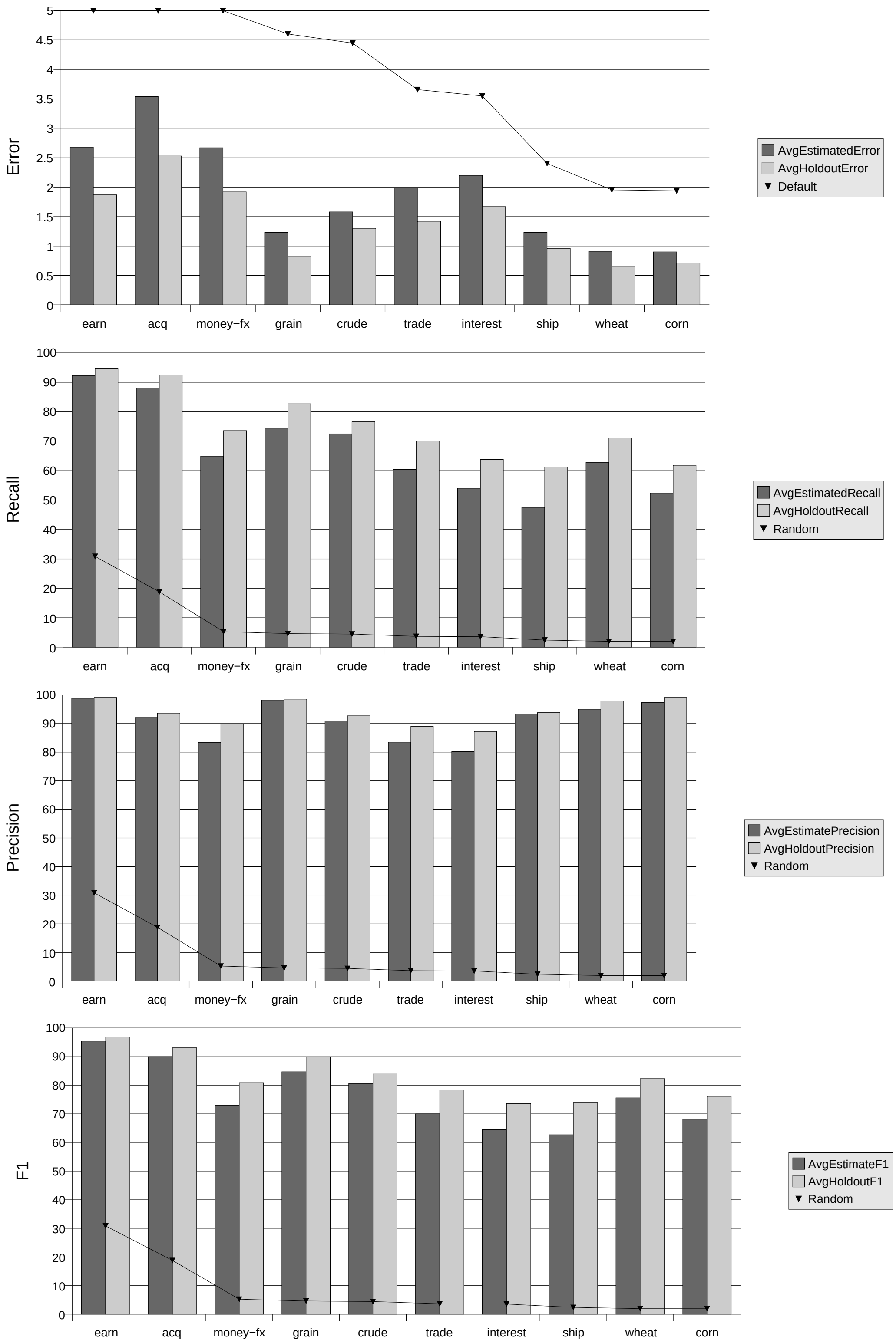


Figure 3: Diagrams comparing average $\xi\alpha$-estimate ($\rho = 1$) of the error, the recall, the precision, and the $F1$-measure with the average true error, true recall, true precision, and true $F1$ measured on a holdout set for the ten most frequent Reuters categories.

| $\rho = 1$ | $Err_{\xi\alpha}(h_{\mathcal{L}})$ | $Err(h_{\mathcal{L}})$ | $Rec_{\xi\alpha}(h_{\mathcal{L}})$ | $Rec(h_{\mathcal{L}})$ | $Prec_{\xi\alpha}(h_{\mathcal{L}})$ | $Prec(h_{\mathcal{L}})$ | $F1_{\xi\alpha}(h_{\mathcal{L}})$ | $F1(h_{\mathcal{L}})$ |
|---|---|---|---|---|---|---|---|---|
| course | $3.48 \pm 0.30$ | $2.43 \pm 0.34$ | $85.9 \pm 1.2$ | $90.7 \pm 1.2$ | $98.0 \pm 0.3$ | $98.0 \pm 0.5$ | $91.6 \pm 0.7$ | $94.2 \pm 0.8$ |
| faculty | $13.40 \pm 0.64$ | $9.80 \pm 0.33$ | $55.7 \pm 2.2$ | $69.2 \pm 1.0$ | $90.5 \pm 1.2$ | $92.2 \pm 1.3$ | $68.9 \pm 1.8$ | $79.0 \pm 0.7$ |
| project | $9.56 \pm 0.46$ | $7.75 \pm 0.34$ | $22.4 \pm 3.6$ | $37.0 \pm 3.1$ | $91.3 \pm 2.5$ | $95.9 \pm 1.2$ | $35.9 \pm 4.8$ | $53.3 \pm 3.1$ |
| student | $10.14 \pm 0.59$ | $7.64 \pm 0.36$ | $82.3 \pm 0.9$ | $87.0 \pm 0.4$ | $90.9 \pm 0.7$ | $93.0 \pm 0.6$ | $86.4 \pm 0.8$ | $89.9 \pm 0.4$ |
| $\rho = 2$ | $Err_{\xi\alpha}(h_{\mathcal{L}})$ | $Err(h_{\mathcal{L}})$ | $Rec_{\xi\alpha}(h_{\mathcal{L}})$ | $Rec(h_{\mathcal{L}})$ | $Prec_{\xi\alpha}(h_{\mathcal{L}})$ | $Prec(h_{\mathcal{L}})$ | $F1_{\xi\alpha}(h_{\mathcal{L}})$ | $F1(h_{\mathcal{L}})$ |
| course | $11.93 \pm 0.47$ | $2.43 \pm 0.34$ | $60.0 \pm 1.8$ | $90.7 \pm 1.2$ | $81.0 \pm 1.2$ | $98.0 \pm 0.5$ | $68.9 \pm 1.4$ | $94.2 \pm 0.8$ |
| faculty | $31.98 \pm 0.78$ | $9.80 \pm 0.33$ | $23.8 \pm 1.7$ | $69.2 \pm 1.0$ | $35.3 \pm 2.2$ | $92.2 \pm 1.3$ | $28.4 \pm 1.9$ | $79.0 \pm 0.7$ |
| project | $14.80 \pm 0.23$ | $7.75 \pm 0.34$ | $2.3 \pm 0.9$ | $37.0 \pm 3.1$ | $8.2 \pm 3.0$ | $95.9 \pm 1.2$ | $3.6 \pm 1.4$ | $53.3 \pm 3.1$ |
| student | $32.42 \pm 0.50$ | $7.64 \pm 0.36$ | $52.5 \pm 1.1$ | $87.0 \pm 0.4$ | $59.8 \pm 0.6$ | $93.0 \pm 0.6$ | $55.9 \pm 0.8$ | $89.9 \pm 0.4$ |

Table 2: Same as table 1, but for the WebKB dataset. The training/test sets contain 2092 examples.

### 6.2 What is the Influence of the Training Set Size?

All results presented so far were for a large training set containing more than 6000 examples. Do the estimator work for smaller training sets as well? Figures 4 to 6 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 similar to $F1$. The top two curves of each graph show the average $\xi\alpha$-estimate and the average holdout-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 holdout-estimate are almost equal. In terms of variance, the training set size has a strong influence on both the holdout-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 holdout-estimator. This confirms that the $\xi\alpha$-estimators have approximately the same variance as holdout testing, but save the cost an additonal test set.

### 6.3 Do the Findings Transfer to Other Text Classification Tasks?

To make sure that the $\xi\alpha$-estimator 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 2) and the Ohsumed data (table 3). For both collections, the results are qualitatively the same as for Reuters.

The conclusion from the experimental findings 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 pessimistic bias and a variance essentially as low as that of holdout-testing using twice as much data.

## 7 Summary and Conclusions

This chapter proposes an approach to estimating the generalization performance of a SVM without any computation intensive resampling. The new 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 estimators delevoped here are the first that address the special measures used to evaluate text classification

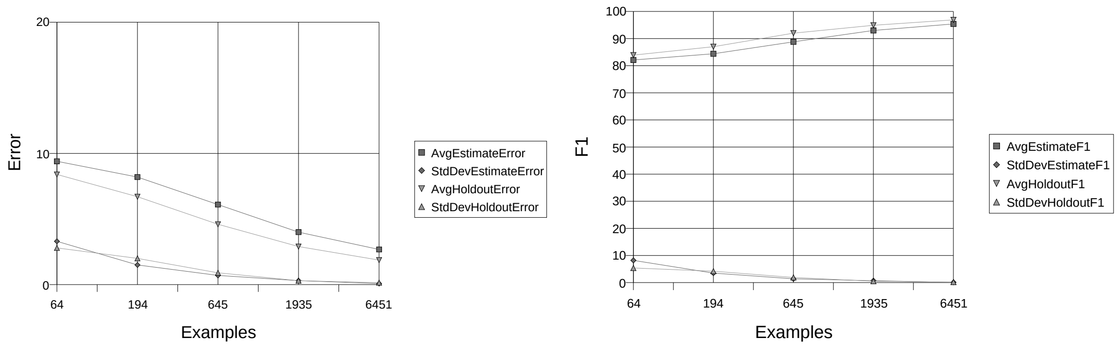


Figure 4: Learning curves for the Reuters category "earn" comparing the $\xi\alpha$-estimator of the error rate (left) and the $F1$ (right) with holdout testing. The x-axis denotes the size of the training set on a log-scale. Each test set for holdout testing contains as many examples as the corresponding 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.

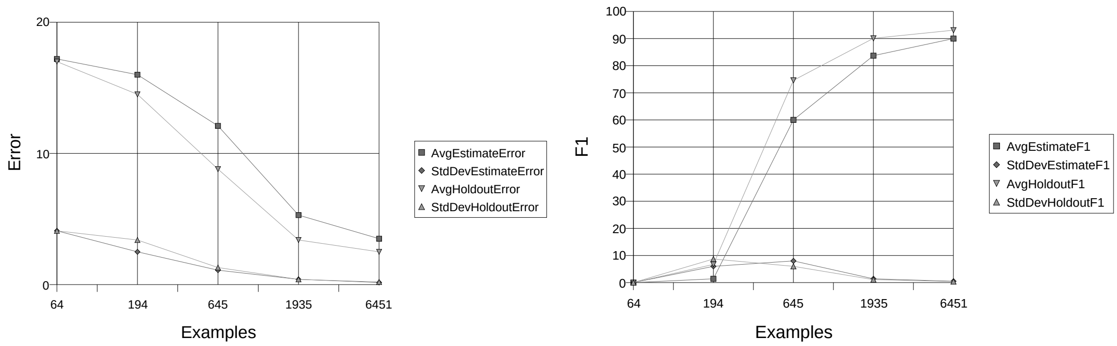


Figure 5: Same as figure 4, but for the Reuters category "acq".

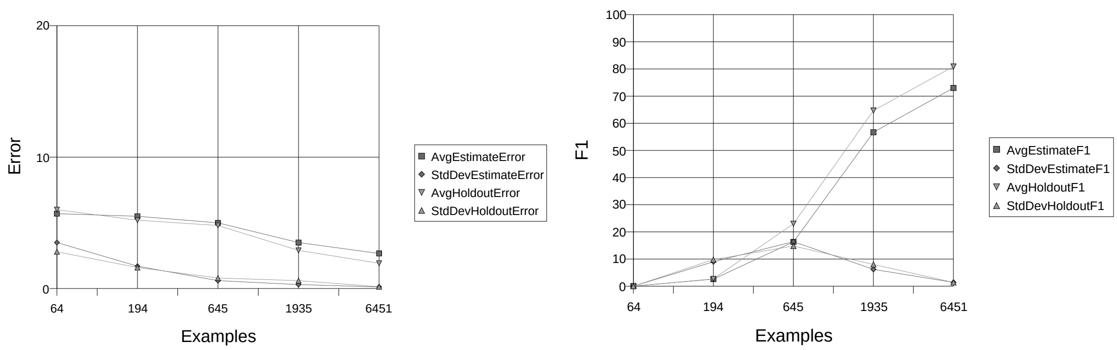


Figure 6: Same as figure 4, but for the Reuters category "money-fx".

| $\rho = 1$ | $Err_{\xi\alpha}(h_{\mathcal{L}})$ | $Err(h_{\mathcal{L}})$ | $Rec_{\xi\alpha}(h_{\mathcal{L}})$ | $Rec(h_{\mathcal{L}})$ | $Prec_{\xi\alpha}(h_{\mathcal{L}})$ | $Prec(h_{\mathcal{L}})$ | $F1_{\xi\alpha}(h_{\mathcal{L}})$ | $F1(h_{\mathcal{L}})$ |
|---|---|---|---|---|---|---|---|---|
| Pathology | $20.96 \pm 0.22$ | $18.56 \pm 0.36$ | $18.9 \pm 1.2$ | $25.8 \pm 1.3$ | $67.0 \pm 1.0$ | $81.2 \pm 2.1$ | $29.5 \pm 1.6$ | $39.1 \pm 1.4$ |
| Cardiovascular | $8.73 \pm 0.24$ | $7.13 \pm 0.15$ | $55.3 \pm 1.5$ | $64.0 \pm 0.8$ | $84.0 \pm 1.0$ | $88.0 \pm 0.8$ | $66.7 \pm 1.3$ | $74.1 \pm 0.7$ |
| Neoplasm | $6.51 \pm 0.15$ | $5.42 \pm 0.14$ | $60.7 \pm 1.0$ | $67.3 \pm 0.8$ | $94.0 \pm 0.4$ | $95.2 \pm 0.6$ | $73.8 \pm 0.7$ | $78.8 \pm 0.6$ |
| Nervous System | $8.41 \pm 0.21$ | $7.31 \pm 0.15$ | $29.0 \pm 2.3$ | $38.3 \pm 1.3$ | $82.0 \pm 1.1$ | $89.6 \pm 1.5$ | $42.8 \pm 2.5$ | $53.6 \pm 1.2$ |
| Immunologic | $6.04 \pm 0.19$ | $5.28 \pm 0.22$ | $37.5 \pm 1.1$ | $45.4 \pm 1.2$ | $86.8 \pm 1.2$ | $90.8 \pm 1.0$ | $52.3 \pm 1.1$ | $60.5 \pm 1.0$ |
| $\rho = 2$ | $Err_{\xi\alpha}(h_{\mathcal{L}})$ | $Err(h_{\mathcal{L}})$ | $Rec_{\xi\alpha}(h_{\mathcal{L}})$ | $Rec(h_{\mathcal{L}})$ | $Prec_{\xi\alpha}(h_{\mathcal{L}})$ | $Prec(h_{\mathcal{L}})$ | $F1_{\xi\alpha}(h_{\mathcal{L}})$ | $F1(h_{\mathcal{L}})$ |
| Pathology | $33.33 \pm 0.50$ | $18.56 \pm 0.36$ | $7.4 \pm 0.6$ | $25.8 \pm 1.3$ | $12.6 \pm 0.9$ | $81.2 \pm 2.1$ | $9.3 \pm 0.7$ | $39.1 \pm 1.4$ |
| Cardiovascular | $15.41 \pm 0.25$ | $7.13 \pm 0.15$ | $34.9 \pm 1.1$ | $64.0 \pm 0.8$ | $51.8 \pm 1.1$ | $88.0 \pm 0.8$ | $41.7 \pm 1.1$ | $74.1 \pm 0.7$ |
| Neoplasm | $11.77 \pm 0.21$ | $5.42 \pm 0.14$ | $42.4 \pm 1.0$ | $67.3 \pm 0.8$ | $67.6 \pm 1.0$ | $95.2 \pm 0.6$ | $52.1 \pm 0.9$ | $78.8 \pm 0.6$ |
| Nervous System | $13.22 \pm 0.19$ | $7.31 \pm 0.15$ | $11.2 \pm 1.0$ | $38.3 \pm 1.3$ | $25.4 \pm 1.8$ | $89.6 \pm 1.5$ | $15.5 \pm 1.3$ | $53.6 \pm 1.2$ |
| Immunologic | $9.37 \pm 0.29$ | $5.28 \pm 0.22$ | $19.8 \pm 1.1$ | $45.4 \pm 1.2$ | $43.6 \pm 1.6$ | $90.8 \pm 1.0$ | $27.3 \pm 1.3$ | $60.5 \pm 1.0$ |

Table 3: Same as table 1, but for the Ohsumed dataset and a training/test set size of 10000.

performance. While they can be used to estimate error rate like standard cross-validation, they can also predict the recall, the precision, and the $F1$.

The theoretical analysis of the estimators shows that they tend to be conservative. This is a desireable 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 holdout estimator which has access to twice as much labelled data. The $\xi\alpha$-estimators are therefor a very promising method for estimating the performance of SVMs on text classification tasks. They make very efficient use of the data and they are computationally very efficient.

Currently, the $\xi\alpha$-estimators are applied to automatic model and parameter selection for text classification [Joachims, 2000]. They can efficiently select between different pre-processing steps (e.g. stemming or no stemming), appropriate kernel parameters, and good values for $C$. Open questions are, whether the bias can be removed with only a modest increase of computational expense. One approach could be to perform actual leave-one-out testing only on those examples for which $\rho\alpha R^2_\Delta + \xi > 1$. It might be possible to integrate these test directly into the optimization process. Finally, it is an open question whether the $\xi\alpha$-estimators work well on other tasks besides text classification.

# 8 Acknowledgements

The financial support of the Deutsche Forschungsgemeinschaft (SFB 475, "Reduction of Complexity for Multivariate Data Structures") is gratefully acknowledged.

# A Notation

| | |
|---|---|
| $\vec{x}_i$ | input patterns |
| $y_i$ | target values (classes) |
| $n$ | number of training examples |
| $N$ | dimensionality of input space |
| $\mathcal{L}$ | learner |
| $\mathcal{H}$ | hypothesis space |
| $h$ | hypothesis from hypothesis space |
| $h_{\mathcal{L}}$ | hypothesis that learner $\mathcal{L}$ returns |
| $R(h)$ | (expected) risk of hypothesis $h$ |
| $R_{emp}(h)$ | empirical risk of hypothesis $h$ on a training sample |
| $\vec{w}$ | weight vector |
| $b$ | constant offset (or threshold) |
| $d$ | VC-dimension |
| $\mathcal{K}$ | Mercer kernel |
| $F$ | feature space |
| $Err$ | Error Rate |
| $Rec$ | Recall |
| $Prec$ | Precision |
| $F1$ | $F1$-measure |
| $\alpha_i$ | Lagrange multiplier |
| $\vec{\alpha}$ | vector of all Lagrange multipliers |
| $\xi_i$ | slack variables |
| $H$ | Hessian of the quadratic program |
| $\mathbf{R}$ | the set of reals |
| $\mathbf{N}$ | the set of natural numbers |
| $(\vec{x}_1 \cdot \vec{x}_2)$ | dot product between patterns $\vec{x}_1$ and $\vec{x}_2$ |
| $\Vert.\Vert$ | $L_2$-norm (Euclidean distance), $\Vert\vec{x}\Vert := \sqrt{(\vec{x} \cdot \vec{x})}$ |
| $e$ | 2.7182818 |
| $exp(a)$ | $e^a$ |
| $\ln$ | logarithm to base $e$ |
| $\log_2$ | logarithm to base 2 |

The difference between a random variable and its realizations is not reflected in the notation. The meaning is either clear from the context, or explicitly explained in the text.

## References


[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.

[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.

[Breiman et al., 1984] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and regression trees.* Wadsworth & Brooks, Pacific Grove.

[Burges and Crisp, 1999] Burges, C. and Crisp, D. (1999). Uniqueness of the SVM solution. In *Conference on Neural Information Processing Systems (NIPS99).*

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. N. (1995). Support–vector networks. *Machine Learning Journal*, 20:273–297.

[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.

[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.

[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.

[Fletcher, 1987] Fletcher, R. (1987). *Practical Methods of Optimization.* Wiley, 2 edition.

[Hoeffding, 1963] Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30.

[Jaakkola and Haussler, 1999] Jaakkola, T. and Haussler, D. (1999). Probabilistic kernel regression models. In *Conference on AI and Statistics.*

[Joachims, 1998] Joachims, T. (1998). 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, 1999] Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11. MIT Press, Cambridge, MA.

[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.

[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.

[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.

[Lachenbruch and Mickey, 1968] Lachenbruch, P. and Mickey, A. (1968). Estimation of error rates in discriminant analysis. *Technometrics*, 10:1–11.

[Lunts and Brailovskiy, 1967] Lunts, A. and Brailovskiy, V. (1967). Evaluation of attributes obtained in statistical decision rules. *Engineering Cybernetics*, 3:98–109.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw Hill, New York.

[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*.

[Rifkin et al., 1999] Rifkin, R., Pontil, M., and Verri, A. (1999). A note on support vector machine degeneracy. In *Conference on Algorithmic Learning Theory (ALT99)*.

[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.

[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.

[Shao and Tu, 1995] Shao, J. and Tu, D. (1995). *The Jackknife and Bootstrap*. Springer, New York.

[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.

[Toussaint and Donaldson, 1970] Toussaint, G. and Donaldson, R. (1970). Algorithms for recognizing contour-traced handprinted characters. *IEEE Transactions on Computers*, 19:541–546.

[Vapnik, 1982] Vapnik, V. (1982). *Estimation of Dependencies Based on Empirical Data.* Springer.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory.* Wiley, Chichester, GB.

[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.