Estimating the Generalization Performance of an SVM Efficiently

Thorsten Joachims JOACHIMS@LS8.INFORMATIK.UNI-DORTMUND.DE
Informatik LS VIII, Universität Dortmund, Baroper Str. 301, 44221 Dortmund, Germany

Abstract

This paper proposes and analyzes an efficient and effective approach for 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 bootstrapping. They can be computed at essentially no extra cost immediately after training a single SVM. Moreover, the estimators developed here address the special performance measures needed for evaluating text classifiers. They can be used not only to estimate the error rate, but also to estimate recall, precision, and F_1. A theoretical analysis and experiments show that the new method can effectively estimate the performance of SVM text classifiers in an efficient way.

1. Introduction

From a practical perspective, learning theory should provide accurate and efficient methods for predicting how well a learner can handle the task at hand. Given a set of training examples, a practitioner will ask how well a particular learner will generalize using this data. With this information it is possible to address the question of selecting between different representation, different models, and different learning parameters. Similarly, it provides the basis for deciding when to apply the learned rule, since real-world tasks usually require a certain minimum performance.

The following presents an approach to answering these questions in the context of text classification with Support Vector Machines (SVMs) (Joachims, 1998; Dumais et al., 1998). The aim is to develop operational performance estimators that are of actual use when applying SVMs. This requires that the estimators are both effective and computationally efficient. Devroye et al. (1996) give an overview of error estimation. While some estimators (e.g., uniform convergence bounds) are powerful theoretical tools, they are of little use in practical applications, since they are too loose. Others (e.g., cross-validation, bootstrapping) give good estimates, but are computationally inefficient. This paper describes new estimators that overcome these problems, extending results of Vapnik (1998) and Jaakkola and Haussler (1999) to general SVMs. The new estimators are both accurate and efficient to compute.

While the results presented here are general enough to apply to arbitrary classification 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. A theoretical analysis as well as experiments show that the new estimators accurately reflect the actual behavior of SVMs on text classification tasks.

2. Text Classification

Text classification is the primary application domain for the methods developed here. The goal of text classification is the automatic assignment of documents to a fixed number of semantic categories. Using a binary classifier for each such category, this leads to the following type of supervised learning problem. Given is an i.i.d. training sample S_n

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

of size n from a fixed but unknown distribution $\Pr(\vec{x}, y)$ describing the learning task. $\vec{x}_i$ represents the document content and $y_i \in \{-1, +1\}$ indicates the class. The learner $\mathcal{L}$ aims to find a decision rule $h_{\mathcal{L}} : \vec{x} \longrightarrow \{-1, +1\}$ based on S_n that classifies new documents as accurately as possible.

Documents, which typically are strings of characters, have to be transformed into a representation suitable for the learning algorithm and the classification task. Here the so called "bag-of-words" representations following the setup of Joachims (1998) is used. Each distinct word corresponds to a feature with its TFIDF score (Salton & Buckley, 1988) as the value. Neither stemming nor stopword-removal are used. To abstract from different document lengths, each document feature vector $\vec{x}_i$ is normalized to unit length.

Unlike for other applications of machine learning, error rate $Err(h)$ alone is not necessarily a good performance measure in text classification, since very often the examples from one class vastly outnumber those from the other class. Instead, scores based on precision and recall are used widely. I define the recall $Rec(h)$ of a decision rule h as the probability that a document with label $y = 1$ is classified correctly. 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. Between high precision and high recall exists a trade-off. A popular method to combine both into a single performance measure is the F_1. It is defined as the geometric mean of precision and recall.

3. Support Vector Machines

The new estimators exploit a particular property of Support Vector Machines that is identified in section 4. SVMs (e.g., Vapnik, 1998) were developed based on the structural risk minimization principle from statistical learning theory. In their basic form, SVMs learn linear decision rules $h(\vec{x}) = sign\{\vec{w} \cdot \vec{x} + b\}$ described by a weight vector $\vec{w}$ and a threshold b. For a given training sample S_n, the SVM finds the hyperplane with maximum soft-margin. Computing this hyperplane is equivalent to solving the following optimization problem (Vapnik, 1998), **(OP1)**:

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

$$\text{subj. to:} \quad \forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i \quad (3)$$

$$\forall_{i=1}^{n} : \xi_i > 0 \quad (4)$$

The constraints (3) require that all training examples are classified correctly up to some slack ξ_i. If a training example lies on the "wrong" side of the hyperplane, the corresponding ξ_i is greater or equal to 1. Therefore, $\sum_{i=1}^{n} \xi_i$ is an upper bound on the number of training errors. The factor C in (2) is a parameter that allows one to trade off training error vs. model complexity. Instead of soving OP1 directly, it is easier to solve the following Wolfe dual of OP1 (Vapnik, 1998), **(OP2)**:

$$\text{minimize:} \; W(\vec{\alpha}) = -\sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j) \quad (5)$$

$$\text{subj. to:} \; \sum_{i=1}^{n} y_i \alpha_i = 0 \quad \wedge \quad \forall_{i=1}^{n} : 0 \leq \alpha_i \leq C \quad (6)$$

In this paper, SVM^{Light} (Joachims, 1999b) is used for computing the solution of OP2[1]. 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 $\vec{\alpha}$ of OP2 the decision rule $h(\vec{x})$ and the solution of OP1 can be computed using $\vec{w} \cdot \vec{x} = \sum_{i=1}^{n} \alpha_i y_i (\vec{x}_i \cdot \vec{x})$, the threshold $b = y_{usv} - \vec{w} \cdot \vec{x}_{usv}$, and the training losses $\xi_i = max(1 - y_i [\vec{w} \cdot \vec{x}_i + b], 0)$. The training example $(\vec{x}_{usv}, y_{usv})$ for calculating b must be an unbounded support vector. While it is highly unlikely in practice that one gets a solution of OP2 with only bounded support vectors, it is theoretically possible (see Burges & 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. If there is at least one unbounded support vector, the solution is called *stable*. While SVMs are an essentially linear method, they can easily be generalized to non-linear decision rules by replacing the inner-products $(\vec{x}_i \cdot \vec{x}_j)$ with a kernel function $\mathcal{K}(\vec{x}_i, \vec{x}_j)$ (Vapnik, 1998) in the formulas above.

4. Efficient Performance Estimators for SVMs

The estimators developed in the following are based on the idea of leave-one-out estimation (Lunts & Brailovskiy, 1967). The leave-one-out estimator of the error rate proceeds as follows. From the training sample $S_n = ((\vec{x}_1, y_1), \cdots, (\vec{x}_n, y_n))$ 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 number of leave-one-out errors $\sum_{i=1}^{n} L(h_{\mathcal{L}}^{\backslash i}(\vec{x}_i), y_i)$ divided by n is the leave-one-out estimate $Err_{loo}^{n}(h_{\mathcal{L}})$ of the generalization error.

A shortcoming of the leave-one-out estimator is its computational inefficiency. The learner needs to be invoked n times for a training set of size n. For most practical problems, this is prohibitively expensive. 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,

[1] www-ai.informatik.uni-dortmund.de/svm_light

but can be applied directly after training the learner on S_n. The inputs to the estimators are the vector $\vec{\alpha}$ solving the dual SVM training problem OP2 and the vector $\vec{\xi}$ from the solution of the primal SVM training problem OP1. Due to this dependence, they will be called $\xi\alpha$-estimators in the following. Both $\vec{\alpha}$ and $\vec{\xi}$ are available at no extra cost after training the SVM.

4.1 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 *For stable soft-margin SVMs, the $\xi\alpha$-estimator of the error rate is*

$$Err^n_{\xi\alpha}(h_{\mathcal{L}})=\frac{d}{n} \quad \textit{with} \quad d = |\{i : (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \quad (7)$$

with ρ equals 2. $\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems OP1 and OP2 on the training set S_n. R^2_Δ is an upper bound on $\mathcal{K}(\vec{x},\vec{x}) - \mathcal{K}(\vec{x},\vec{x}')$ for all $\vec{x},\vec{x}'$.

$R^2_\Delta = 1$ for the linear kernel $\mathcal{K}(\vec{x}_i,\vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$ on normalized document vectors. 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. 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 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 $S_n^{\backslash i}$, 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 *The number of leave-one-out errors $\sum_{i=1}^n L(h_{\mathcal{L}}^{\backslash i}(\vec{x}_i), y_i)$ of stable soft-margin SVMs on a training set S_n is bounded by*

$$\sum_{i=1}^n L(h_{\mathcal{L}}^{\backslash i}(\vec{x}_i), y_i) \leq |\{i : (2\alpha_i R^2 + \xi_i) \geq 1\}| \qquad (8)$$

$\vec{\alpha}$ and $\vec{\xi}$ are the solution of optimization problems OP1 and OP2 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 (sketch, more details in (Joachims, 1999a)) An error on a left-out example $(\vec{x}_t, y_t)$ occurs when at the solution of

$$W_t(\vec{\alpha}^t) = max_{\vec{0} \leq \vec{\alpha}^t \leq \vec{C}} 1^T \vec{\alpha}^t - \frac{1}{2}\vec{\alpha}^{tT} H^t \vec{\alpha}^t \wedge \vec{y}^{tT}\vec{\alpha}^t = 0 \quad (9)$$

where $\vec{\alpha}^t$, $\vec{y}^t$, and H^t have the t-th example removed, the expression $y_t \left[\sum_{i \neq t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b^t\right] > 0$ is false. What follows in this proof are conditions for when this expression must be true based on the soft-margin SVM solution

$$W(\vec{\alpha}) = max_{\vec{0} \leq \vec{\alpha} \leq \vec{C}} 1^T \vec{\alpha} - \frac{1}{2}\vec{\alpha}^T H \vec{\alpha} \wedge \vec{y}^T \vec{\alpha} = 0 \quad (10)$$

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 $y_t \sum_{i \neq t} y_i \alpha_i^t \mathcal{K}(\vec{x}_t, \vec{x}_i) = y_t \sum_{i \neq t} y_i \alpha_i \mathcal{K}(\vec{x}_t, \vec{x}_i) = 1 > 0$.

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 $W(.)$.

$$\beta_i = \begin{cases} \alpha_i^t & if \quad \alpha_i^t = 0 \vee \alpha_i^t = C \\ \alpha_i^t - y_i y_t \nu_i & if \quad i \in SV^t \\ \alpha_i & if \quad i = t \end{cases} \qquad (11)$$

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 $W(.)$. It is now possible to transform (10) into

$$\begin{aligned} \alpha_t y_t \left[\sum_{i \in SV^t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b^t\right] = \\ -W(\vec{\beta}) + W_t(\vec{\alpha}^t) - \frac{1}{2}\alpha_t^2 \mathcal{K}(\vec{x}_t, \vec{x}_t) + \alpha_t \qquad (12) \\ -\frac{1}{2}\sum_{i \in SV^t} \sum_{j \in SV^t} \nu_i \nu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \alpha_t \sum_{i \in SV^t} \nu_i \mathcal{K}(\vec{x}_i, \vec{x}_t) \end{aligned}$$

A similar construction produces a feasible point $\vec{\gamma}$ of $W_t(.)$ based on the solution $\vec{\alpha}$ of $W(.)$.

$$\gamma_i = \begin{cases} \alpha_i & if \quad \alpha_i = 0 \vee \alpha_i = C \\ \alpha_i + y_i y_t \mu_i & if \quad i \in SV \setminus \{t\} \end{cases} \qquad (13)$$

SV is the set of indices corresponding to support vectors not at the upper bound for the solution $\vec{\alpha}$ (that is $0 < \alpha_i < C$). $SV\backslash\{t\}$ excludes the index t corresponding to the left-out example. μ_i is chosen non-negative for all $i \in SV\backslash\{t\}$ such that $\sum_{i \in SV\backslash\{t\}} \mu_i = \alpha_t$ 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 $W_t(.)$. Exploiting that $W(\vec{\alpha}) \geq W(\vec{\beta})$ by definition, and $W_t(\vec{\alpha}^t) \geq W^t(\vec{\gamma})$, substitution into (12) leads to

$$\alpha_t y_t \left[\sum_{i \in SV^t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b^t \right] \geq \alpha_t y_t \left[\sum_{i \in SV\backslash\{t\}} \alpha_i y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b \right]$$
$$- \frac{1}{2} \sum_{i \in SV\backslash\{t\}} \sum_{j \in SV\backslash\{t\}} \mu_i \mu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \alpha_t \sum_{i \in SV\backslash\{t\}} \mu_i \mathcal{K}(\vec{x}_i, \vec{x}_t) \quad (14)$$
$$- \frac{1}{2} \sum_{i \in SV^t} \sum_{j \in SV^t} \nu_i \nu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) + \alpha_t \sum_{i \in SV^t} \nu_i \mathcal{K}(\vec{x}_i, \vec{x}_t) \quad (15)$$

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 $\alpha_t \sum_{i \in SV\backslash\{t\}} \mu_i \mathcal{K}(\vec{x}_i, \vec{x}_t)$. Furthermore, $\frac{1}{2} \sum_{i \in SV\backslash\{t\}} \sum_{j \in SV\backslash\{t\}} \mu_i \mu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \leq \frac{1}{2}\alpha_t^2 R^2$ and $\frac{1}{2} \sum_{i \in SV^t} \sum_{j \in SV^t} \nu_i \nu_j \mathcal{K}(\vec{x}_i, \vec{x}_j) \leq \frac{1}{2}\alpha_t^2 R^2$, 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. Using these inequalities and dividing by α_t, it is possible to write $y_t \left[\sum_{i \in SV^t} \alpha_i^t y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b^t\right] \geq y_t \left[\sum_{i \in SV\backslash\{t\}} \alpha_i y_i \mathcal{K}(\vec{x}_t, \vec{x}_i) + b\right] - \alpha_t R^2$. 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 one can see that a leave-one-out error can occur only when $2\alpha_t R^2 \geq 1$. Note that $\xi_i = 0$ for support vectors.

Case $\alpha_t = C$*:* Example $(\vec{x}_t, y_t)$ is a bounded support vector. The argumentation follows that in the case of unbounded support vectors. The only difference is 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, leading to the condition $2\alpha_t R^2 + \xi_t \geq 1$. ▮

The idea of connecting the leave-one-out error with properties of the solution vector $\vec{\alpha}$ goes back to Vapnik (1998, pages 418-421). Unlike the work presented here, Vapnik's result is limited to the case where (a) the training data is separable, (b) the special case of hyperplanes passing through the origin, and (c) it is used to derive bounds on the expected error, not estimators. Jaakkola and Haussler (1999) present a generalized bound for inseparable data that is similar to that of Lemma 1. 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 by Wahba (1999) and Opper and Winther (in press). An additional difference is that their approach requires computing the inverse for part of the Hessian making it computationally more expensive.

While Lemma 1 is valid for all kernel functions that return positive values, it is tightest when the minimum value is zero. The following lemmashows that this can always be achieved.

Lemma 2 *The soft-margin SVM is invariant under addition of a real value c to the kernel function.*

The proof is given in (Joachims, 1999a). The previous two lemmas complete the tools needed to characterize the bias of the $\xi\alpha$-estimator of the error rate.

Theorem 1 *The $\xi\alpha$-estimator of the error rate is pessimistically biased in the following sense*

$$\mathcal{E}(Err_{\xi\alpha}^n(h_{\mathcal{L}})) \geq \mathcal{E}(Err^{n-1}(h_{\mathcal{L}})) \quad (16)$$

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 A theorem of Lunts and Brailovskiy (1967) 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. Lemma 1 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. Bounds in (Devroye & Wagner, 1976) require that the probability $\Pr(h_{\mathcal{L}}^n(\vec{x}) \neq h_{\mathcal{L}}^{\backslash i}(\vec{x}))$ is small. For SVMs this quantity depends on the learning task. Bounds on the variability of the leave-one-out estimator presented by Kearns and Ron (1997) are independent of the learning task. Nevertheless, their bounds would be too loose to be of practical importance. Therefore, the variability of the $\xi\alpha$-estimator will be assessed empirically in section 5.

4.2 Recall, Precision, and F_1

With similar arguments as for the error rate, one can derive $\xi\alpha$-estimators of the recall, the precision, and the $F1$.

Definition 2 *For stable soft-margin SVMs, the $\xi\alpha$-estimators of the recall, the precision, and the $F1$ are*

$$Rec^n_{\xi\alpha}(h_{\mathcal{L}}) = 1 - \frac{d_+}{n_+} \qquad (17)$$

$$Prec^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{n_+ - d_+}{n_+ - d_+ + d_-} \qquad (18)$$

$$F1^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{2\,n_+ - 2\,d_+}{2\,n_+ - d_+ + d_-} \qquad (19)$$

$$\textit{using} \quad d_+ = |\{i : y_i = 1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \qquad (20)$$

$$d_- = |\{i : y_i = -1 \wedge (\rho\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \qquad (21)$$

$$n_+ = |\{i : y_i = 1\}| \qquad (22)$$

with ρ equals 2. $\vec{\alpha}$ and $\vec{\xi}$ are the solution of OP1 and OP2 on the training set S_n. R^2_Δ is an upper bound on $\mathcal{K}(\vec{x},\vec{x}) - \mathcal{K}(\vec{x},\vec{x}')$ for all $\vec{x},\vec{x}'$.

d_+ (d_-) is the number of positive (negative) 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. Using the same arguments as in Lemma 1, d_+ (d_-) is an upper bound on the number of positive (negative) examples that produce a leave-one-out error.

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 (i.e. the elements of the contingency tables) are averaged before the function is applied. This removes the artifacts discussed above. The micro-expected value $\mathcal{E}_{micro}(f(X))$ of a function $f(x)$ is defined as $\mathcal{E}_{micro}(f(X)) = f(\mathcal{E}(X))$. The random variable X can be a vector. The bias of the $\xi\alpha$-estimators in terms of a micro-expectation is characterized by the following theorem.

Theorem 2 *The $\xi\alpha$-estimators of the recall, the precision, and the $F1$ are pessimistically biased in the following sense:*

$$\mathcal{E}_{micro}(Rec^n_{\xi\alpha}(h_{\mathcal{L}})) \leq \mathcal{E}_{micro}(Rec^{n-1}(h_{\mathcal{L}})) \qquad (23)$$

$$\mathcal{E}_{micro}(Prec^n_{\xi\alpha}(h_{\mathcal{L}})) \leq \mathcal{E}_{micro}(Prec^{n-1}(h_{\mathcal{L}})) \qquad (24)$$

$$\mathcal{E}_{micro}(F1^n_{\xi\alpha}(h_{\mathcal{L}})) \leq \mathcal{E}_{micro}(F1^{n-1}(h_{\mathcal{L}})) \qquad (25)$$

The proof is given in (Joachims, 1999a). The theorem shows that the $\xi\alpha$-estimates are "on average" lower than the true recall, precision, and $F1$ in terms of a micro-average. A similar result for the strong (macro) definition of bias requires further assumptions.

5. 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. Due to space constraints, here only the results for Reuters are discussed in detail. The results for WebKB and Ohsumed can be found in (Joachims, 1999a).

The Reuters-21578 dataset[2] 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.

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 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 (14) and (15) 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. Many support vectors are almost orthogonal. Consequently the expressions in (14) and (15) can be kept close to zero for the linear kernel and suitable $\vec{\nu}, \vec{\mu}$, leading to a $\xi\alpha$-estimate with $\rho \approx 1$ instead of $\rho = 2$.

Unless noted otherwise, the following results are averages over 10 random test/training splits. Training and test set are designed to be of equal size to be able to compare variance estimates. 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). Note that $R_\Delta = 1$, since document vectors are normalized to unit length.

[2] www.research.att.com/~lewis/reuters21578.html

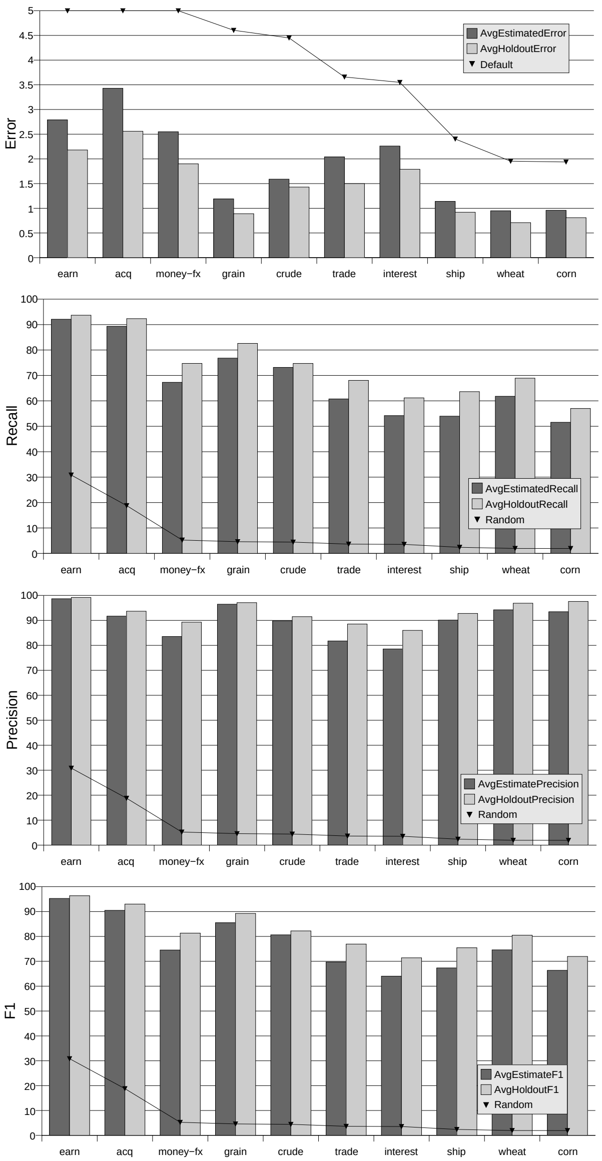

Figure 1. Diagrams comparing average $\xi\alpha$-estimate ($\rho = 1$) of the error, the recall, the precision, and the $F1$-measure with the average true error, true recall, true precision, and true $F1$ measured on a holdout set for the ten most frequent Reuters categories.

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

Figure 1 illustrates the results for the Reuters dataset with $\rho = 1$. The findings are as expected. The $\xi\alpha$-estimators slightly overestimate the true error and underestimate precision, recall, and $F1$. Nevertheless, they accurately reflect the relative performance between categories. Table 1 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.

5.2 What is the Influence of the Training Set Size?

The results presented so far were for a large training set. Do the estimators work for smaller training sets as well? Figure 2 shows 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. 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 "acq" and "money-fx" with small training sets, the SVM behaves almost like the default classifier that assigns all test examples to the more populous class. 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.

5.3 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 also conducted for the WebKB and the Ohsumed data (reported in (Joachims, 1999a)). For both collections, the results are qualitatively the same as for Reuters. 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 relatively low variance.

Table 1. Table comparing average $\xi\alpha$-estimates with the average true performance 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.

$\rho = 1$	$Err_{\xi\alpha}(h_{\mathcal{L}})$	$Err(h_{\mathcal{L}})$	$Rec_{\xi\alpha}(h_{\mathcal{L}})$	$Rec(h_{\mathcal{L}})$	$Prec_{\xi\alpha}(h_{\mathcal{L}})$	$Prec(h_{\mathcal{L}})$	$F1_{\xi\alpha}(h_{\mathcal{L}})$	$F1(h_{\mathcal{L}})$
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

6. Summary and Conclusions

This paper proposed 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 bootstrapping, since they can be computed immediately after training a single SVM. Moreover, the estimators developed here address the special measures used to evaluate text classification performance.

The 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. 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 labeled data. The $\xi\alpha$-estimators are therefore a suitable method for estimating the performance of SVMs on text classification tasks. They make efficient use of the data and they are computationally efficient.

Currently, the $\xi\alpha$-estimators are applied to model selection for text classification (Joachims, 2000). They can effectively select between different preprocessing steps (e.g. stemming or no stemming), kernel parameters, and good values for C. Similarly, they can be used to detect concept drift (Klinkenberg & Joachims, 2000). An open questions is whether the bias can be removed with only a modest increase of computational expense. One approach could be to integrate actual leave-one-out testing into the optimization process considering only those examples for which $\rho\alpha R_{\Delta}^2 + \xi > 1$.

Acknowledgments

This work was supported by the DFG Collaborative Research Center on Complexity Reduction in Multivariate Data (SFB475).

References

Burges, C., & Crisp, D. (1999). Uniqueness of the SVM solution. *Proceedings of the Twelfth Conference on Neural Information Processing Systems*. Cambridge, MA: MIT Press.

Devroye, L., Györfi, L., & Lugosi, G. (1996). *A probabilistic theory of pattern recognition*. Springer.

Devroye, L., & Wagner, T. (1976). A distribution-free performance bound in error estimation. *IEEE Transactions on Information Theory*, *22*, 586–587.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. *Proceedings of the Eighth Conference on Information and Knowledge Management* (pp. 148–155). New York: ACM Press.

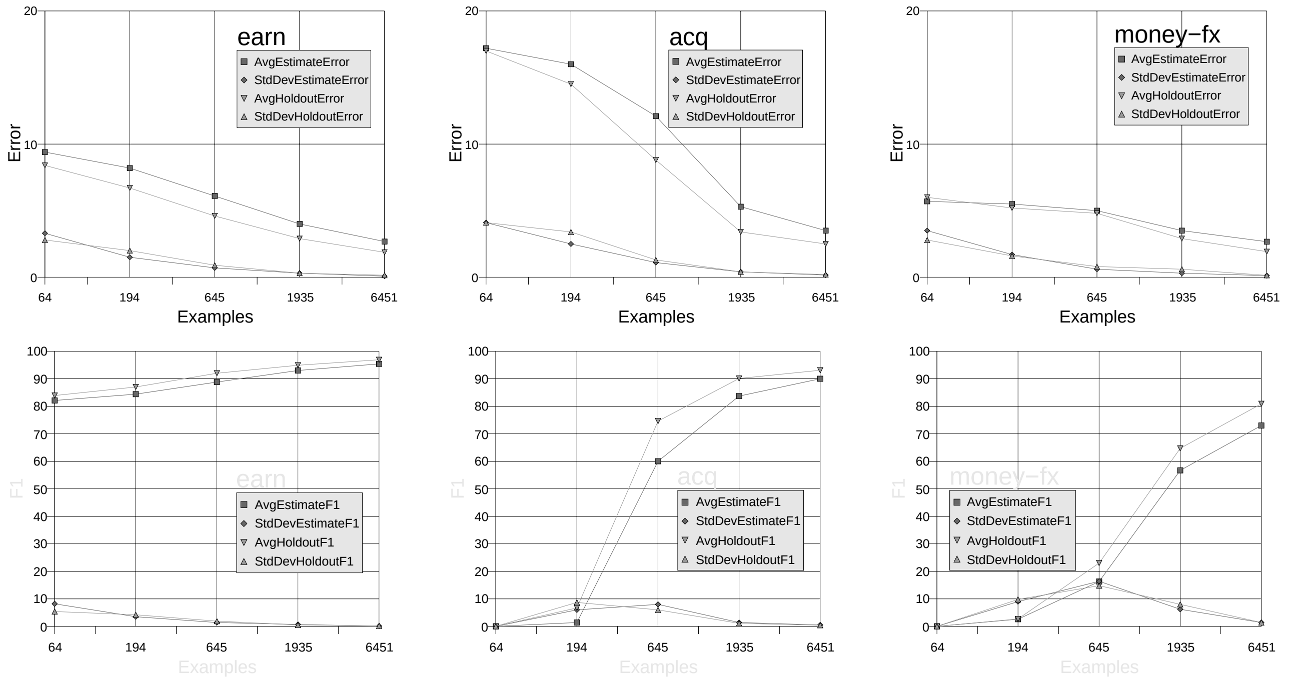

Figure 2. Learning curves for the Reuters categories "earn", "acq", and "money-fx" comparing the $\xi\alpha$-estimator of the error rate (upper row) and the $F1$ (lower row) with holdout testing. The x-axis denotes the size of the training set. Each test set for holdout testing contains as many examples 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.

Jaakkola, T., & Haussler, D. (1999). Probabilistic kernel regression models. *Proceedings of the Seventh Workshop on AI and Statistics.* San Francisco: Morgan Kaufman.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Proceedings of the Tenth European Conference on Machine Learning* (pp. 137 – 142). Berlin: Springer.

Joachims, T. (1999a). *Estimating the generalization performance of a SVM efficiently* (LS VIII-Report 25). Universität Dortmund, Germany.

Joachims, T. (1999b). Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods - Support vector learning.* Cambridge, MA: MIT Press.

Joachims, T. (2000). *Efficient model selection for text classification with support vector machines* (LS VIII-Report 26). Universität Dortmund, Germany.

Kearns, M., & Ron, D. (1997). Algorithmic stability and sanity-check bounds for leave-one-out cross validation. *Proceedings of the Tenth Conference on Computational Learning Theory* (pp. 152–162). New York: ACM Press.

Klinkenberg, R., & Joachims, T. (2000). Detecting concept drift with support vector machines. *Proceedings of the Seventeenth International Conference on Machine Learning.* San Francisco: Morgan Kaufman.

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

Opper, M., & Winther, O. (in press). Gaussian process classification and SVM: Mean field results and leave-one-out estimator. In P. Bartlett, B. Schölkopf, D. Schuurmans, & A. Smola (Eds.), *Large margin classifiers.* Cambridge, MA: MIT Press.

Rifkin, R., Pontil, M., & Verri, A. (1999). A note on support vector machine degeneracy. *Proceedings of the Tenth Conference on Algorithmic Learning Theory.* Berlin: Springer.

Salton, G., & Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management*, *24*, 513–523.

Vapnik, V. (1998). *Statistical learning theory.* Chichester, UK: Wiley.

Wahba, G. (1999). Support vector machines, reproducing kernel hilbert spaces, and randomized gacv. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods - Support vector learning.* Cambridge, MA: MIT Press.