Stopping Conditions for Exact Computation of Leave-One-Out Error in Support Vector Machines

Vojtěch Franc[1] VOJTECH.FRANC@FIRST.FRAUNHOFER.DE
Pavel Laskov[1,2] PAVEL.LASKOV@FIRST.FRAUNHOFER.DE
Klaus-Robert Müller[1,3] KLAUS@FIRST.FRAUNHOFER.DE

[1]Fraunhofer Institute FIRST, Kekulestr. 7, 12489 Berlin, Germany
[2]University of Tuebingen, Wilhelm Schickard Instutute for Computer Science, Sand 13, 72070 Tuebingen, Germany
[3]Technical University of Berlin, Dept. of Computer Science, Franklinstr. 28/29, 10587 Berlin, Germany

Abstract

We propose a new stopping condition for a Support Vector Machine (SVM) solver which precisely reflects the objective of the Leave-One-Out error computation. The stopping condition guarantees that the output on an intermediate SVM solution is identical to the output of the optimal SVM solution with one data point excluded from the training set. A simple augmentation of a general SVM training algorithm allows one to use a stopping criterion equivalent to the proposed sufficient condition. A comprehensive experimental evaluation of our method shows consistent speedup of the exact LOO computation by our method, up to the factor of 13 for the linear kernel. The new algorithm can be seen as an example of constructive guidance of an optimization algorithm towards achieving the best attainable expected risk at optimal computational cost.

1. Introduction

The interrelation between a computational complexity and a generalization ability of learning algorithms has seldom been considered in machine learning. Since the solutions to a majority of learning problems are obtained by iterative optimization algorithms, solution accuracy plays an important role in the estimation of expected risk (Bartlett & Mendelson, 2006). In practice, the available computational resources necessitate a tradeoff between approximation accuracy determined by the choice of a class of functions, estimation error determined by a finite set of examples, and an optimization error determined by the accuracy of a solver attainable within a given time budget (Bottou & Bousquet, 2008).

The asymptotic analysis in (Bottou & Bousquet, 2008) provides upper bounds on the time required to reach a certain expected risk by a given algorithm. From the practical point of view, it is desirable to have *constructive* influence over a learning algorithms, by choosing its parameters, such as e.g. learning rate or stopping conditions, to reach the best attainable expected risk. The present contribution provides an example of such a constructive mechanism by developing optimal stopping conditions for SVM training using a particular estimator of an expected risk – the leave-one-out (LOO) error. Although exact computation of a LOO error is hardly used for large-scale learning due to its computational burden, our method is feasible for "small-scale" learning with "expensive" data (e.g. in bioinformatics or finance), especially when accurate estimation of expected risk is required.

The LOO is known to provide an unbiased estimator of the generalization error (Lunts & Brailovskiy, 1967). The naive computation of the LOO error, i.e. by explicit relearning after exclusion of each single example, is in all but the simplest cases impractical. The problem of speeding up a computation of a LOO error has received significant attention. The following approaches exist:

- LOO bounds provide an estimate of the LOO error given an optimal solution of the SVM training problem ((Joachims, 2000; Vapnik & Chapelle, 2000; Jaakkola & Haussler, 1999; Zhang, 2001)). These bounds are computationally efficient but imprecise. In practice, if an accurate estimate of the classification accuracy is needed, exact computation of the LOO error is unavoidable.
- Incremental SVM (Cauwenberghs & Poggio, 2000) allows one to exactly determine for each candidate

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

training point – *after* obtaining the optimal SVM solution – whether or not it will be a LOO error. This approach avoids explicit re-training, but incremental unlearning of points is complicated and requires special organization of matrix operations (Laskov et al., 2007).

- Loose stopping conditions based on the ε-KKT allow one to speed up the LOO computation *before* obtaining an optimal solution. Such methods, (e.g. (Lee & Lin, 2000; Martin et al., 2004)) use fairly simple heuristics, but lack theoretical justification that would connect the ε to a precision of the LOO computation. As it is illustrated in the examples in Section 2, these methods can also be imprecise.

In this contribution we propose a new stopping condition for an SVM solver which *precisely reflects* the objective of the LOO error computation. Our main result, given in Theorem 1, provides a sufficient condition for which the output on an intermediate SVM solution is identical to the output of the optimal SVM solution with one data point excluded from the training set. Although this sufficient condition cannot be computed in practice, we propose a simple augmentation of a general SVM training algorithm which allows one to use a stopping criterion equivalent to the proposed sufficient condition.

2. Leave-One-Out Error Estimate For Support Vector Machines Classifier

Let $\mathcal{X}$ be a set of inputs and $\mathcal{Y} = \{-1, +1\}$ a set of labels of an analyzed object. Let further $\mathcal{T}_{XY} = \{(x_1, y_1), \ldots, (x_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$ be a finite training set i.i.d. sampled from unknown $P(x, y)$. The goal is to learn a classifier $f\colon \mathcal{X} \to \mathcal{Y}$ minimizing the probability of misclassification $R[f] = \int V(y, f(x))dP(x, y)$ where $V(y, y') = 1$ for $y \neq y'$ and $V(y, y') = 0$ otherwise.

The SVMs represent the input states in the Reproducing Kernel Hilbert Space (RKHS) via a map $\mathbf{\Phi}\colon \mathcal{X} \to \mathcal{H}$ which is implicitly defined by a kernel function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (Schölkopf & Smola, 2002). The classifier is assumed to be linear, i.e., $f(x; \boldsymbol{w}, b) = \langle \boldsymbol{w}, \mathbf{\Phi}(x)\rangle + b$, where $\boldsymbol{w} \in \mathcal{H}$, $b \in \mathbb{R}$ are unknown parameters and $\langle\cdot,\cdot\rangle$ denotes an inner product in RKHS. Because $R[f]$ cannot be minimized directly due to the unknown $P(x, y)$, the SVMs replace $R[f]$ by a regularized risk its minimization leads to

$$(\boldsymbol{w}^*, b^*) = \operatorname*{argmin}_{\boldsymbol{w}\in\mathcal{H}, b\in\mathbb{R}} \left(\frac{1}{2}\|\boldsymbol{w}\|^2_{\mathcal{H}} + C \sum_{i\in\mathcal{I}} \hat{V}(y_i, f(x_i; \boldsymbol{w}, b)) \right) \tag{1}$$

where $C \in \mathbb{R}^+$ is a regularization constant, $\hat{V}(y_i, f(x_i; \boldsymbol{w},)) = \max(0, 1 - y_i f(x_i; \boldsymbol{w}, b))$ is a convex piece-wise linear approximation of $V(y_i, f(x_i; \boldsymbol{w}, b))$ and $\mathcal{I} = \{1, \ldots, m\}$. By the Representer theorem (Schölkopf & Smola, 2002), the optimal SVM classifier $f(x; \boldsymbol{w}^*, b^*)$ can be expressed in the form

$$f(x; \boldsymbol{\alpha}, b) = \begin{cases} +1 & \text{if} \quad \sum\limits_{i\in\mathcal{I}} \alpha_i y_i k(x, x_i) + b \geq 0\,, \\ -1 & \text{if} \quad \sum\limits_{i\in\mathcal{I}} \alpha_i y_i k(x, x_i) + b < 0\,, \end{cases} \tag{2}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^T \in \mathbb{R}^m$, $b \in \mathbb{R}$. Substituting (2) to (1) allows to find the optimal SVM classifier by solving a convex QP task

$$(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*) = \operatorname*{argmin}_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi})\in\mathcal{A}} F(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \tag{3}$$

where the convex objective function reads

$$F(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) = \frac{1}{2} \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{I}} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + C \sum_{i\in\mathcal{I}} \xi_i\,,$$

and the convex feasible set $\mathcal{A}$ contains all $(\boldsymbol{\alpha} \in \mathbb{R}^m, b \in \mathbb{R}, \boldsymbol{\xi} \in \mathbb{R}^m)$ satisfying

$$\begin{aligned} y_i \left(\sum_{j\in\mathcal{I}} \alpha_j y_j k(x_i, x_j) + b \right) &\geq 1 - \xi_i\,, \quad i \in \mathcal{I}\,, \\ \xi_i &\geq 0\,, \qquad i \in \mathcal{I}\,. \end{aligned}$$

Minimizing the regularized risk (1) (or QP task (3) respectively) allows to find parameters $(\boldsymbol{\alpha}, b)$ of the SVM classifier provided the hyper-parameters, i.e., the kernel function k and the regularization constant C, are known. This is not the case in practice and the hyper-parameters (C, k) must be optimized as well. A common approach is to select the best (C, k) from a given finite set Θ by minimizing some performance measure. The set Θ is usually created by reasonably discretizing the hyper-parameters space. As the performance measure, the LOO error $R_{LOO}[f]$ is a common choice.

Let $(\boldsymbol{\alpha}^*(r), b^*(r), \boldsymbol{\xi}^*(r))$ denote the optimal solution of the primal QP task (3) with r-th example removed from the training set which is equivalent to solving the task

$$(\boldsymbol{\alpha}^*(r), b^*(r), \boldsymbol{\xi}^*(r)) = \operatorname*{argmin}_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi})\in\mathcal{A}(r)} F(\boldsymbol{\alpha}, b, \boldsymbol{\xi})\,, \tag{4}$$

where $\mathcal{A}(r) = \mathcal{A} \cap \{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \mid \alpha_r = 0\}$, i.e., $\mathcal{A}(r)$ denotes the original feasible set $\mathcal{A}$ enriched by an additional constraint $\alpha_r = 0$. The LOO error estimator is defined as

$$R_{LOO}[f(\cdot; \boldsymbol{\alpha}^*, b^*)] = \frac{1}{m} \sum_{r\in\mathcal{I}} V(y_r, f(x_r; \boldsymbol{\alpha}^*(r), b^*(r)))\,. \tag{5}$$

The major practical disadvantage of the LOO error is its high computational cost. A naive approach to compute LOO requires solving m different QP tasks (4). In some

cases, however, the value $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ can be immediately derived from the optimal solution $(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*)$ computed from the entire training set. Table 1 summarizes the known sufficiency checks; the implication 1 is generally known and the implications 2, 3 are due to (Joachims, 2000).

1. If $\alpha_r^* = 0$ then $y_r = f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$.
2. If $y_r \neq f(x_r; \boldsymbol{\alpha}^*, b^*)$ then $y_r \neq f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$.
3. Let R^2 be an upper bound on $k(x,x) - k(x,x')$, $\forall x, x' \in \mathcal{X}$, and let $(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*)$ be a stable solution which means that there exist at least one $i \in \mathcal{I}$ such that $0 < \alpha_i^* < C$. In this case, if $2\alpha_r^* R^2 + \xi_r^* < 1$ then $y_r = f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$.

Table 1. Sufficiency checks for computing $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ directly from $(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*)$.

A portion of the training examples for which the sufficiency checks apply depends on the problem at hand (for empirical study see (Martin et al., 2004)). (Joachims, 2000) proposed using the sufficiency checks to compute an upper bound on the LOO error called $\xi\alpha$-estimator. It has been empirically shown, that in general the $\xi\alpha$-estimator is not sufficiently precise for the hyper-parameter tuning (Duan et al., 2003). Algorithm 1 shows a standard procedure of computing the LOO error exactly with the use of the sufficiency checks to reduced the number of cases when the solution of the QP task (4) is required.

Algorithm 1 Computation of the LOO Error

1: Solve the QP task (3) to obtain $(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*)$.
2: Apply the sufficiency checks from Table 1 to compute $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ from $(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*)$.
3: For examples unresolved in Step 2 solve the QP task (4) to obtain $(\boldsymbol{\alpha}^*(r), b^*(r))$.
4: Compute the LOO error by (5) using $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$, $r \in \mathcal{I}$ obtained in Steps 2 and 3.

Algorithm 1 requires the use of an optimization method solving the QP tasks (3) and (4) exactly, i.e., producing the optimal solutions. Although such optimization methods exist, they are applicable only for very small problems. In practice, the QP tasks are solved only approximately via their dual representation which is more suitable for optimization due to a simpler feasible set. In particular, the minimizer $\boldsymbol{\alpha}^*$ of the primal QP task (3) can be equivalently computed by solving the dual QP task

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha} \in \mathcal{B}} \left(\sum_{i \in \mathcal{I}} \alpha_i - \frac{1}{2} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right), \tag{6}$$

where $\mathcal{B}$ is a convex feasible set which contains all $\boldsymbol{\alpha} \in \mathbb{R}^m$ satisfying

$$\sum_{i \in \mathcal{I}} \alpha_i y_i = 0\,, \quad \text{and} \quad 0 \leq \alpha_i \leq C\,, i \in \mathcal{I}\,.$$

We will use $G(\boldsymbol{\alpha})$ to denote the objective function of (6). Having $\boldsymbol{\alpha}^*$ computed, the remaining primal variables $(b^*, \boldsymbol{\xi}^*)$ can be obtained easily from the Karush-Kuhn-Tucker (KKT) optimality conditions (e.g., (Boyd & Vandenberghe, 2004)). Similarly, the minimizer $\boldsymbol{\alpha}^*(r)$ of the QP task (4) is obtained by solving the dual

$$\boldsymbol{\alpha}^*(r) = \operatorname*{argmax}_{\boldsymbol{\alpha} \in \mathcal{B}(r)} G(\boldsymbol{\alpha})\,, \tag{7}$$

where $\mathcal{B}(r) = \mathcal{B} \cap \{\boldsymbol{\alpha} \mid \alpha_r = 0\}$ and the primal variables $(b^*(r), \boldsymbol{\xi}^*(r))$ can be again obtained by the KKT conditions. From the optimization point of view, the QP tasks (6) and (7) are equivalent since the latter can be converted to the former simply by excluding the r-th variable. Thus we now concentrate only on the optimization of the QP task (6).

Algorithm 2 Commonly used iterative QP solver

1: Initialize $t := 0$ and $\boldsymbol{\alpha}^{(t)} \in \mathcal{B}$.
2: $t := t + 1$.
3: Update $\boldsymbol{\alpha}^{(t-1)} \rightarrow \boldsymbol{\alpha}^{(t)}$, i.e., find $\boldsymbol{\alpha}^{(t)} \in \mathcal{B}$ such that $G(\boldsymbol{\alpha}^{(t-1)}) < G(\boldsymbol{\alpha}^{(t)})$.
4: If $\boldsymbol{\alpha}^{(t)}$ satisfies the ε-KKT conditions (8) halt otherwise go to 2.

A framework of a commonly used QP solver optimizing (6) describes Algorithm 2. Among the most popular methods which fit to the framework of Algorithm 2 belong the Sequential Minimal Optimizer (SMO) (Platt, 1998), SVM ^{light} (Joachims, 1998) and other decomposition methods (e.g. (Vapnik, 1995; Osuna et al., 1997)). All these solvers iteratively increase the dual criterion $G(\boldsymbol{\alpha})$ until the solution satisfies stopping conditions. A relaxed version of the KKT optimality conditions is the most frequently used stopping criterion: let $\varepsilon \geq 0$ be a prescribed number and $\nabla_i(\boldsymbol{\alpha}) = 1 - y_i \sum_{j \in \mathcal{I}} \alpha_j y_j k(x_i, x_j)$; then a vector $\boldsymbol{\alpha} \in \mathcal{B}$ satisfies the relaxed KKT conditions (e.g. (Keerthi et al., 2001)) if there exist $b \in \mathbb{R}$ such that

$$\begin{array}{rcll} \nabla_i(\boldsymbol{\alpha}) + by_i & \leq & \varepsilon\,, & \text{if} \quad \alpha_i = 0\,, \\ -\nabla_i(\boldsymbol{\alpha}) + by_i & \leq & \varepsilon\,, & \text{if} \quad \alpha_i = C\,, \\ |\, -\nabla_i(\boldsymbol{\alpha}) + by_i \,| & \leq & \varepsilon\,, & \text{if} \quad 0 < \alpha_i < C\,. \end{array} \tag{8}$$

The tightness of the stopping conditions (8) is controlled by $\varepsilon \geq 0$; hereafter we will refer to (8) as the ε-KKT conditions. An advantage of the ε-KKT is their simplicity and a low computational overhead: $O(m)$ operations since $\nabla_i(\boldsymbol{\alpha})$ is usually available during the course of the

QP solver. A disadvantage of the ε-KKT conditions is a tricky choice of ε. Provided $\varepsilon = 0$, the solution $\boldsymbol{\alpha}$ satisfying the ε-KKT conditions is guaranteed to be optimal. The practically applicable QP solvers, however, are guaranteed to halt in a finite number of iterations only for $\varepsilon > 0$. A typically used value is $\varepsilon = 0.001$, e.g., in software packages SVM^{light} ((Joachims, 1998)) or `svmlib` ((Chang & Lin, 2001)). To our knowledge, there is no theoretical result connecting $\varepsilon > 0$ to the value of $R_{LOO}[f(\cdot; \boldsymbol{\alpha}^*, b^*)]$ which is the only desired outcome of the entire computation.

We will illustrate the impact of ε when the LOO error estimator is used for a model selection. Let Θ be a given finite set of hyper-parameters $\boldsymbol{\theta} = (C, k)$. Let $R_{LOO}(\boldsymbol{\theta}, \varepsilon)$ denote the LOO error estimate computed for given $\boldsymbol{\theta}$ using Algorithm 1 with a QP solver in Algorithm 2. Thus the estimated LOO error $R_{LOO}(\theta, \varepsilon)$ is a function of both the hyper-parameters $\boldsymbol{\theta}$ and ε. For a fixed value $\varepsilon > 0$, the model selection produces the hyper-parameters

$$\boldsymbol{\theta}(\varepsilon) = \operatorname*{argmin}_{\boldsymbol{\theta} \in \Theta} R_{LOO}(\boldsymbol{\theta}, \varepsilon) \; . \tag{9}$$

Figure 1 plots the behavior of $R_{LOO}(\boldsymbol{\theta}(\varepsilon), \varepsilon)$ and $R_{LOO}(\boldsymbol{\theta}(10^{-4}), \varepsilon)$, as well as the cost of the LOO error computation as a function of ε for three datasets selected from the IDA repository (cf. Section 5).

The "golden truth" expected risk is given by the left-most plots in the graphs (using $\varepsilon = 10^{-4}$ for both model selection and risk estimation). The dashed line representing $R_{LOO}(\boldsymbol{\theta}(10^{-4}), \varepsilon)$ shows that the expected risk is slightly overestimated provided we use high accuracy for model selection and variable accuracy for risk estimation. The solid line representing $R_{LOO}(\boldsymbol{\theta}(\varepsilon), \varepsilon)$ shows that a low-accuracy LOO computation used in model selection eventually results in overfitting, as a model is selected that grossly underestimates the expected risk. Interestingly, both plots coincide until a certain breakdown point beyond which the low-accuracy LOO estimation runs aground. The breakdown point varies between 0.001 and 0.1 depending on a dataset. This suggests that a commonly used $\varepsilon = 0.001$ is a reasonable setting to obtain an accurate estimate. It is, however, clear from the timing plots that knowing the right accuracy could significantly lower the computational cost.

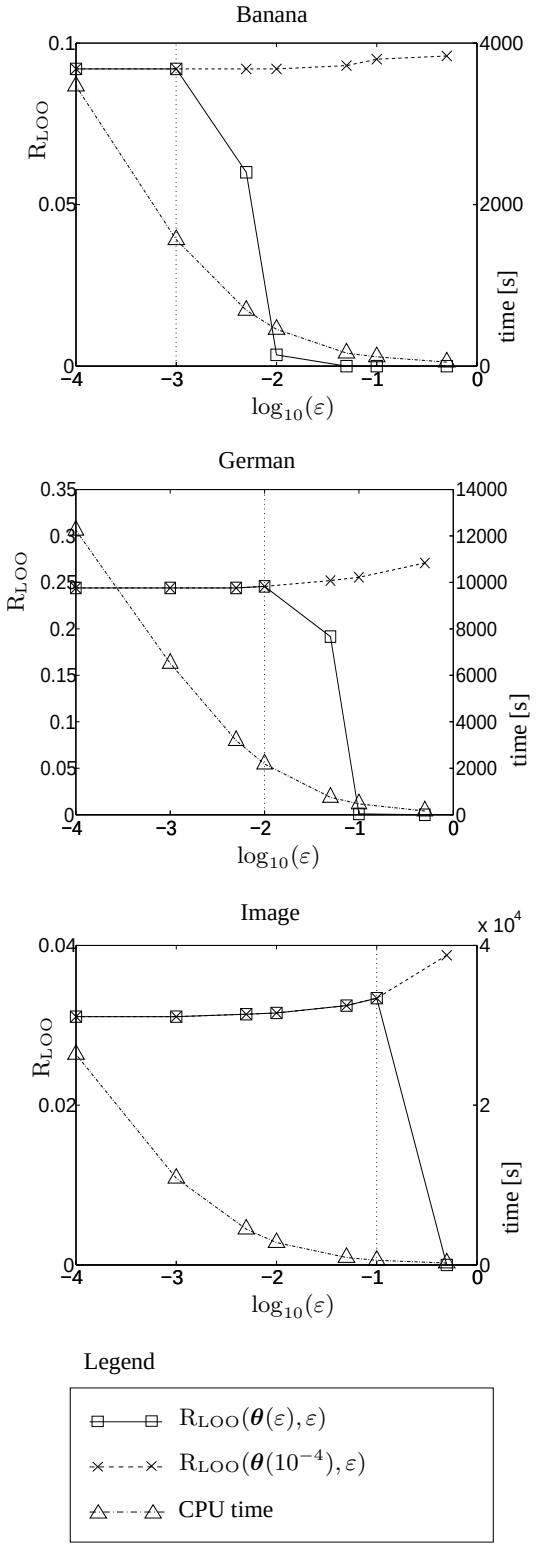

Figure 1. Influence of the parameter ε of the ε-KKT conditions on the LOO error estimate and the required computational time for three data sets (Banana, German and Image) selected from IDA repository.

3. Exact Computation of the LOO Error

In this section we show that a response of the optimal classifier $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$, required when the LOO error is being computed, can be obtained without the need to solve the QP task (4) (or its dual (7)) optimally. Let us define

three convex sets

$$\begin{aligned}\mathcal{A}_+(r) &= \mathcal{A}(r) \cap \Big\{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \mid \sum_{i\in\mathcal{I}} \alpha_i y_i k(x_i, x_r) + b > 0\Big\},\\ \mathcal{A}_0(r) &= \mathcal{A}(r) \cap \Big\{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \mid \sum_{i\in\mathcal{I}} \alpha_i y_i k(x_i, x_r) + b = 0\Big\},\\ \mathcal{A}_-(r) &= \mathcal{A}(r) \cap \Big\{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \mid \sum_{i\in\mathcal{I}} \alpha_i y_i k(x_i, x_r) + b < 0\Big\}.\end{aligned} \tag{10}$$

Notice, that to compute $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ we do not necessarily need to know the optimal $(\boldsymbol{\alpha}^*(r), b^*(r), \boldsymbol{\xi}^*(r))$ but it suffices to determine whether $(\boldsymbol{\alpha}^*(r), b^*(r), \boldsymbol{\xi}^*(r))$ belongs to $\mathcal{A}_+(r) \cup \mathcal{A}_0(r)$ or to $\mathcal{A}_-(r)$. Our method is based on a simple observation which can be formally stated by the following theorem:

Theorem 1 *For any $(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}}) \in \mathcal{A}(r)$ which satisfy*

$$F(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}}) < \min_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \in \mathcal{A}_0(r)} F(\boldsymbol{\alpha}, b, \boldsymbol{\xi})\,, \tag{11}$$

the equation $f(x_r; \hat{\boldsymbol{\alpha}}, \hat{b}) = f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ holds.

Proof 1 *We proof Theorem 1 by transposition: we show that $f(x_r; \hat{\boldsymbol{\alpha}}, \hat{b}) \neq f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ implies the assumption (11) is violated. Without loss of generality let $f(x_r; \hat{\boldsymbol{\alpha}}, \hat{b}) = +1$ and $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r)) = -1$. Let us define three vectors*

$$\hat{\boldsymbol{\theta}} = \begin{pmatrix} \hat{\boldsymbol{\alpha}} \\ \hat{b} \\ \hat{\boldsymbol{\xi}} \end{pmatrix}, \quad \boldsymbol{\theta}^*(r) = \begin{pmatrix} \boldsymbol{\alpha}^*(r) \\ b^*(r) \\ \boldsymbol{\xi}^*(r) \end{pmatrix}, \quad \boldsymbol{\theta}_0 = \begin{pmatrix} \boldsymbol{\alpha}_0 \\ b_0 \\ \boldsymbol{\xi}_0 \end{pmatrix}.$$

With a slight abuse of notation, we will handle F as a function of a single argument $\boldsymbol{\theta} \in \mathbb{R}^{2m+1}$. Then the assumptions $f(x_r; \hat{\boldsymbol{\alpha}}, \hat{b}) = +1$ and $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r)) = -1$ is equivalent to $\hat{\boldsymbol{\theta}} \in \mathcal{A}_+(r) \cup \mathcal{A}_0(r)$ and $\boldsymbol{\theta}^*(r) \in \mathcal{A}_-(r)$. From (10) it follows that for any $\hat{\boldsymbol{\theta}} \in \mathcal{A}_+(r) \cup \mathcal{A}_0(r)$ and $\boldsymbol{\theta}^*(r) \in \mathcal{A}_-(r)$ there exists $\tau \in [0, 1]$ such that $\boldsymbol{\theta}_0 = ((1-\tau)\hat{\boldsymbol{\theta}} + \tau\boldsymbol{\theta}^*(r)) \in \mathcal{A}_0(r)$. Since F is convex, $\tau \in [0, 1]$ and $F(\boldsymbol{\theta}^*) \leq F(\hat{\boldsymbol{\theta}})$ we can write

$$\begin{aligned} F(\boldsymbol{\theta}_0) &\leq (1-\tau)F(\hat{\boldsymbol{\theta}}) + \tau F(\boldsymbol{\theta}^*(r)) \\ &\leq \max\big\{F(\hat{\boldsymbol{\theta}}), F(\boldsymbol{\theta}^*(r))\big\} \quad = \quad F(\hat{\boldsymbol{\theta}})\,, \end{aligned}$$

which shows that there exist $\boldsymbol{\theta}_0 \in \mathcal{A}_0(r)$ such that $F(\boldsymbol{\theta}_0) \leq F(\hat{\boldsymbol{\theta}})$. Using the original notation, this is equivalent to $F(\boldsymbol{\alpha}_0, b_0, \boldsymbol{\xi}_0) \leq F(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}})$. However, this contradicts the assumption (11) which was to be shown.

By Theorem 1, any triplet $(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}}) \in \mathcal{A}(r)$ satisfying the condition (11) determines a classifier $f(x; \hat{\boldsymbol{\alpha}}, \hat{b})$ which has the same response on the input x_r as the optimal classifier $f(x; \boldsymbol{\alpha}^*(r), b^*(r))$. From a practical point of view, this result cannot be used directly due to the unknown value of the right hand side of the inequality (11), i.e.,

$$\min_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \in \mathcal{A}_0(r)} F(\boldsymbol{\alpha}, b, \boldsymbol{\xi})\,. \tag{12}$$

The problem (12) is a convex QP task its dual reads

$$\boldsymbol{\beta}^*(r) = \max_{\boldsymbol{\beta} \in \mathcal{B}_0(r)} \left(\sum_{i\in\mathcal{I}} \beta_i - \frac{1}{2} \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{I}} \beta_i \beta_j y_i y_i k'(x_i, x_j) \right), \tag{13}$$

where $k'(x_i, x_j) = k(x_i, x_j) - k(x_r, x_i) - k(x_r, x_j) - k(x_r, x_r)$ and $\mathcal{B}_0(r)$ is a convex feasible set which contains all $\boldsymbol{\beta} \in \mathbb{R}^m$ satisfying

$$0 \leq \beta_i \leq C\,, i \in \mathcal{I} \setminus \{r\}\,, \quad \text{and} \quad \beta_r = 0\,.$$

We will use $H(\boldsymbol{\beta})$ to denote the objective function of (13). By the weak duality theorem, the inequality $F(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \geq H(\boldsymbol{\beta})$ holds for any $(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \in \mathcal{A}_0(r)$ and $\boldsymbol{\beta} \in \mathcal{B}_0(r)$. This allows us to derive the following useful corollary:

Corollary 1 *For any $(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}}) \in \mathcal{A}_0(r)$ and $\boldsymbol{\beta} \in \mathcal{B}_0(r)$ which satisfy*

$$F(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}}) < H(\hat{\boldsymbol{\beta}})\,, \tag{14}$$

the equation $f(x_r; \hat{\boldsymbol{\alpha}}, \hat{b}) = f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ holds.

Notice, that the condition (14) is satisfiable except for a very rare degenerate cases. It is easy to show, that if the condition (14) is not satisfiable then the error estimate $V(y_r; f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ is unstable anyway since there exists an optimal classifier $f(x; \boldsymbol{\alpha}^*(r), b^*(r))$ its separating hyperplane passes through the tested point x_r.

4. Algorithm

A direct application of Corollary 1 would require solving a mixed set of one quadratic and many linear inequalities. We are not aware of any simple and efficient algorithm to solve such task. Instead, we show how to use Corollary 1 to derive a novel stopping condition for a standard iterative QP solver (cf. Algorithm 2).

Algorithm 3 Proposed QP solver

1: Initialize $t := 0$, $\boldsymbol{\alpha}^{(t)} \in \mathcal{B}(r)$ and $\boldsymbol{\beta}^{(t)} \in \mathcal{B}_0(r)$.
2: $t := t + 1$.
3: Update $\boldsymbol{\alpha}^{(t-1)} \rightarrow \boldsymbol{\alpha}^{(t)}$, i.e., find $\boldsymbol{\alpha}^{(t)} \in \mathcal{A}(r)$ such that $G(\boldsymbol{\alpha}^{(t-1)}) < G(\boldsymbol{\alpha}^{(t)})$.
4: If $\boldsymbol{\alpha}^{(t)}$ satisfies the ε-KKT conditions then halt otherwise continue to Step 5.
5: For fixed $\boldsymbol{\alpha}^{(t)}$ compute feasible $b^{(t)}$ and $\boldsymbol{\xi}^{(t)}$ minimizing $F(\boldsymbol{\alpha}^{(t)}, b, \boldsymbol{\xi})$.
6: Update $\boldsymbol{\beta}^{(t-1)} \rightarrow \boldsymbol{\beta}^{(t)}$, i.e., find $\boldsymbol{\beta}^{(t)} \in \mathcal{B}_0(r)$ such that $H(\boldsymbol{\beta}^{(t-1)}) < H(\boldsymbol{\beta}^{(t)})$.
7: If $F(\boldsymbol{\alpha}^{(t)}, b^{(t)}, \boldsymbol{\xi}^{(t)}) < H(\boldsymbol{\beta}^{(t)})$ holds then halt otherwise go to Step 2.

The proposed method is described by Algorithm 3 which, compared to a standard QP solver, involves three additional Steps 5, 6 and 7. In Step 5, the algorithm computes the

primal variables $(\boldsymbol{\xi}^{(t)}, b^{(t)})$ minimizing $F(\boldsymbol{\alpha}^{(t)}, b^{(t)}, \boldsymbol{\xi}^{(t)})$ which amounts to a simple optimization problem since $\boldsymbol{\alpha}^{(t)}$ is known. In Step 6, the algorithm maximizes the auxiliary criterion $H(\boldsymbol{\beta}^{(t)})$ w.r.t. $\boldsymbol{\beta}^{(t)}$. Finally, in Step 7, the algorithm checks whether $H(\boldsymbol{\beta}^{(t)})$ has become greater than $F(\boldsymbol{\alpha}^{(t)}, b^{(t)}, \boldsymbol{\xi}^{(t)})$; as soon as this occurs the algorithm halts since $f(x_r; \boldsymbol{\alpha}^{(t)}, b^{(t)}) = f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ is guaranteed according to Corollary 1. The ε-KKT conditions are retained in Step 3 of Algorithm 3 since the condition (14) need not be satisfiable in general.

The proposed Algorithm 3 is intended to be used for computation of $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$, i.e., it is called in Step 4 of Algorithm 1 calculating the LOO error, as a replacement for the standard QP solver. In terms of accuracy of computing the LOO error, the proposed algorithm cannot perform worse than the standard one. If the ε-KKT conditions are satisfied earlier than the proposed stopping condition then both the solvers find an identical classifier. In the opposite case, however, the response of the classifier found by the proposed algorithm is guaranteed to be optimal. Albeit the proposed algorithm provides a theoretical guarantee for the found solution to be optimal, from the practical point of view both the algorithms will produce an identical LOO error estimate for a sufficiently low ε. We will empirically show, however, that the proposed algorithm is numerically more efficient though it optimizes two QP tasks simultaneously compared to the standard approach. The higher efficiency is achieved by the proposed stopping condition which is often satisfied earlier than the ε-KKT condition.

To increase the numerical performance we also implemented the following simple efficiency test. The proposed algorithm is not applied on a single example but rather on a set of examples which cannot be resolved by the sufficiency checks. We experimentally observed, that the efficiency of the proposed algorithm can be reliably estimated from a few examples. This allows us to switch to using the standard QP solver when the efficiency of the proposed algorithm is low. The efficiency test, implemented in Step 4 of Algorithm 1, works as follows: We apply the proposed Algorithm 3 on the first M examples. Let M_{Prec} denote the number of examples for which Algorithm 3 halt in Step 7 (i.e., the proposed stopping condition was applied). If $M_{Prec}/M < 0.5$ we switch from using Algorithm 3 to using the standard Algorithm 2. We empirically found $M = 10$ to be a good choice number in all our experiments.

5. Experiments

In this section, we experimentally evaluate the proposed method for computing the LOO error compared to the standard approach on the datasets from the IDA benchmark repository [1].

The standard approach computes the LOO error using the procedure described by Algorithm 1. An iterative QP solver with the ε-KKT conditions (Algorithm 2) is called whenever the solution of the QP task is required. In particular, we used the Improved SMO algorithm (Keerthi et al., 2001) to implement the QP solver. In addition, we implemented α-seeding approach (DeCoste & Wagstaff, 2000) which re-uses the solution $\boldsymbol{\alpha}^*$ (obtained in Step 1 of Algorithm 1) to efficiently set up the initial solution of the QP solver (initialization of $\boldsymbol{\alpha}^{(0)}$ in Step 1 of Algorithm 2).

The proposed approach uses the same procedure for computing the LOO error except for a different QP solver used in Step 4 of Algorithm 1. As the QP solver, we applied the proposed Algorithm 3 which involves optimization of the QP tasks $G(\boldsymbol{\alpha}^{(t)})$ w.r.t. $\boldsymbol{\alpha}^{(t)} \in \mathcal{B}(r)$ and $H(\boldsymbol{\beta}^{(t)})$ w.r.t. $\boldsymbol{\beta}^{(t)} \in \mathcal{B}_0(r)$ required in Step 3 and Step 6, respectively. We again used the Improved SMO algorithm to optimize $G(\boldsymbol{\alpha}^{(t)})$ w.r.t. $\boldsymbol{\alpha}^{(t)} \in \mathcal{B}(r)$ and its straightforward modification to optimize $H(\boldsymbol{\beta}^{(t)})$ w.r.t. $\boldsymbol{\beta}^{(t)} \in \mathcal{B}_0(r)$ ($\mathcal{B}_0(r)$ does not contain the equality constraint thus a single variable can be updated).

The experiments were carried out in Matlab 6 environment runnig on the Linux machine with the AMD K8 2.2GHz processor. Algorithms 1,2 and 3 were implemented in C.

The IDA repository consists of 13 artificial and real-world binary classification problems collected from UCI, DELVE and STATLOG repositories (c.f. (Rätsch et al., 2001)). For each dataset, there are 100 random realizations of training and testing set (except for Image and Splice sets, where it is 20). The training parts of the first 5 realizations are used for model selection. The best hyper-parameters (C, k) were selected from a finite set Θ by minimizing an average LOO error $\hat{R}_{LOO}$. The average LOO error $\hat{R}_{LOO}$ is computed over the 5 realizations.

We considered two separate model selection problems for the linear and the RBF kernel. In the case of the liner kernel, the model was select from $\Theta = \{C \mid C = 10^i, i = -2, \ldots, 2\} \times \{k \mid k(x, x') = \langle x, x' \rangle\}$ and, in the case of the RBF kernel $\Theta = \{C \mid C = 10^{\frac{i}{7}\log(500)}, i = 0, \ldots, 7\} \times \{k \mid k(x, x') = \exp(-2^{\frac{i}{7}11}\|x - x'\|^2), i = 0, \ldots, 7\}$. Having the model selected, the classifier is trained for all 100 realizations of the training sets and the testing error is computed on the corresponding testing set. The reported testing errors $\hat{R}_{TST}$ are averages accompanied with the standard deviations computed over the 100 realizations.

Table 2 shows the average LOO errors $\hat{R}_{LOO}$ and the testing errors $\hat{R}_{TST}$ for the best selected models. We experimentally verified, that both the standard and the proposed

[1] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm

approach yielded identical LOO errors since a low value of $\varepsilon = 0.001$ was used in the ε-KKT conditions. Therefore the errors listed in Table 2 apply for both the approaches. We also found that the classification errors for the RBF kernel are very similar to the errors reported in (Rätsch et al., 2001) for the SVM classifier with the RBF kernel tuned by the 5-fold cross-validation. Interestingly, the linear kernel achieves in some cases comparable performance as the more complex RBF kernel. We can also observe, that the average LOO errors $\hat{R}_{LOO}$ for the linear kernel are very good estimators of the testing errors $\hat{R}_{TST}$.

Table 3 summarizes the numerical efficiency of the proposed approach and the standard one. The efficiency was measured in terms of the computational time and the number of kernel evaluations. The reported $Time$ is the overall computational time spent by a given algorithm to calculate all the LOO errors needed for the model selection. E.g., in the case of the RBF kernel it was necessary to compute $5 \times 64 = 320$ LOO error estimates (5 stands for the number of the training set realizations and 64 is the cardinality of Θ). Similarly, the number of kernel evaluations $KerEval$ is the overall value normalized to the number of training data m, i.e., $KerEval$ is the number of columns of the kernel matrix. In the case of the standard approach, we listed the absolute values of $Time$ and $KerEval$. In the case of the proposed approach, we listed the gained speed up computed as the ratio $Standard/Proposed$. The last column of Table 3 contains the value $Prec$ being the percentage of the cases when the proposed stopping condition was satisfied earlier then the ε-KKT conditions, i.e., in $Prec$ cases the computed LOO error is theoretically guaranteed to be optimal.

It can be seen, that the proposed method was never slower (up to the rounding error in computing the speed up) than the standard algorithm both in terms of the computational time and the kernel evaluations. A higher performance was achieved for the linear kernel compared to the RBF kernel. For the linear kernel, the proposed approach was on average 4 times faster than the standard approach. The best performance was achieved for the Image dataset when the speed up was nearly 13. For RBF kernel, the average speed up was slightly higher than 2, and, in the best case the speed up was 5 for the Banana dataset. It shows that while the efficiency gained for the RBF kernel is only moderate, in the case of the linear kernel it is much appealing.

6. Conclusions

The new stopping conditions for an SVM solver proposed in this contribution allow to determine an optimal solution accuracy needed for exact computation of a LOO error. Our new algorithm allows one to significantly reduce complexity of the LOO error computation without a risk of overfitting due to imprecise optimization. Our experiments on 13 datasets from the IDA repository achieved the average speedup of 2 to 4 times and the maximal speedup of up to the factor of 13.

	Classification performance			
	Linear kernel		RBF kernel	
	$\hat{\mathrm{R}}_{\mathrm{LOO}}$	$\hat{\mathrm{R}}_{\mathrm{TST}}$	$\hat{\mathrm{R}}_{\mathrm{LOO}}$	$\hat{\mathrm{R}}_{\mathrm{TST}}$
Banana	41.40	47.80 ($\pm$4.58)	8.55	10.43 ($\pm$0.44)
Breast	27.20	29.00 ($\pm$4.83)	23.00	26.06 ($\pm$4.91)
Diabetis	22.05	23.44 ($\pm$1.70)	21.50	23.27 ($\pm$1.65)
Flare	32.85	32.33 ($\pm$1.82)	32.31	34.04 ($\pm$2.04)
German	24.91	24.06 ($\pm$2.22)	23.71	23.61 ($\pm$2.23)
Heart	14.24	15.22 ($\pm$3.22)	13.53	15.55 ($\pm$3.36)
Image	15.48	15.34 ($\pm$0.84)	2.94	3.15 ($\pm$0.63)
Ring.	23.45	24.59 ($\pm$0.67)	1.10	1.60 ($\pm$0.11)
Splice	15.36	16.20 ($\pm$0.59)	10.60	10.95 ($\pm$0.64)
Thyroid	8.43	10.16 ($\pm$2.60)	2.29	4.87 ($\pm$2.28)
Titanic	21.20	23.01 ($\pm$4.62)	15.47	23.99 ($\pm$3.47)
Twono.	2.50	2.90 ($\pm$0.27)	2.25	2.59 ($\pm$0.18)
Wave.	10.90	12.95 ($\pm$0.54)	8.85	10.50 ($\pm$0.43)

Table 2. Classification performance of the best models selected by minimizing the LOO error estimate.

These results demonstrate the importance of investigating relationships between the optimization accuracy and the expected risk estimation in machine learning, as suggessed by recent work (Bartlett & Mendelson, 2006; Bottou & Bousquet, 2008). To our knowledge, the new algorithm is the first theoretically justified constructive instrument to guide an optimization algorithm – for the particular case of the SVM QP solver and the LOO error – towards achieving the best attainable expected risk at optimal computational cost. Future work should explore more general mechanisms of relating parameters of optimization algorithms deployed in machine learning with the estimation of expected risk.

Acknowledgements

VF was supported by Marie Curie Intra-European Fellowship grant SCOLES (MEIF-CT-2006-042107). This work was also supported by *Bundesministerium für Bildung und Forschung* under the project REMIND (FKZ 01-IS07007A).

References

Bartlett, P., & Mendelson, S. (2006). Empirical minimization. *Probability Theory and Related Fields, 135*, 311–334.

Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. In *Advances in neural information processing systems*, vol. 20. Cambridge, MA: MIT Press. to appear.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimiza-*

Linear Kernel

	Standard approach		Proposed approach		
	Time [s]	KerEval $\times 10^6$	Time speedup	KerEval speedup	Prec [%]
Banana	728.0	109.5	9.9	12.3	56.9
Breast	128.7	34.6	2.0	2.1	64.2
Diabetis	3705.3	432.1	6.3	7.2	93.1
Flare	61.5	4.7	1.0	1.1	59.3
German	47603.8	3685.7	4.2	4.4	82.0
Heart	280.7	90.1	1.9	2.1	83.1
Image	114373.9	4793.8	12.9	14.9	98.3
Ring.	11632.8	1536.0	2.8	3.0	91.8
Splice	41072.8	2222.1	1.5	1.5	74.5
Thyroid	21.6	8.2	3.2	5.2	66.0
Titanic	0.8	0.1	1.0	1.0	17.6
Twono.	52.8	8.3	1.9	2.2	89.1
Wave.	3422.8	503.3	2.4	2.6	90.7

RBF Kernel

	Standard approach		Proposed approach		
	Time [s]	KerEval $\times 10^6$	Time speedup	KerEval speedup	Prec [%]
Banana	1565.1	222.4	5.1	6.1	81.9
Breast	265.9	74.0	1.5	1.6	55.6
Diabetis	3163.6	382.3	1.4	1.5	54.8
Flare	2263.0	189.1	2.2	2.4	50.3
German	6513.7	549.4	1.1	1.1	31.3
Heart	59.5	20.7	1.1	1.2	50.3
Image	10852.7	514.5	2.9	3.1	75.3
Ring.	277.2	47.4	1.4	1.4	39.1
Splice	6360.3	453.8	1.0	1.0	14.6
Thyroid	10.3	3.3	1.5	2.4	80.3
Titanic	25.5	7.9	5.7	2.7	50.2
Twono.	175.3	36.8	1.0	1.0	22.2
Wave.	345.6	59.4	1.1	1.1	30.9

Table 3. Efficiency of computing the LOO error for the standard and the proposed approach

tion. Cambridge University Press.

Cauwenberghs, G., & Poggio, T. (2000). Incremental and decremental support vector machine learning. *NIPS* (pp. 409–415). MIT Press.

Chang, C., & Lin, C. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

DeCoste, D., & Wagstaff, K. (2000). Alpha seeding for support vector machines. *Int. Conf. on Knowledge Disc. and Data Mining*.

Duan, K., Keerti, S., & Poo, A. (2003). Evaluation of simple performance measures for tuning SVM hyperparamaters. *Neurocomputing*, *15*, 487–507.

Jaakkola, T., & Haussler, D. (1999). Probabilistic kernel regression models. *Conference on AI and Statistics*. Morgan Kaufmann.

Joachims, T. (1998). Making large-scale support vector machine learning practical. In Schölkopf et B. al. (Ed.), *Advances in kernel methods: Support vector machines*. MIT Press, Cambridge, MA.

Joachims, T. (2000). Estimating the generalization performance of a SVM efficiently. *ICML*. San Francisco, CA.

Keerthi, S., Shevade, S., Bhattacharyya, C., & Murthy, K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, *13*, 637–649.

Laskov, P., Gehl, C., Krüger, S., & Müller, K.-R. (2007). Incremental support vector learning: analysis, implementation and applications. *Journal of Machine Learning Research*, *7*, 1900–1936.

Lee, J., & Lin, C. (2000). *Automatic model selection for support vector machines* (Technical Report). Dept. of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan,.

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

Martin, M., Keerthi, S., Ong, C., & DeCoste, D. (2004). An efficient method for computing leave-one-out error in support vector machines with gaussian kernels. *IEEE TNN*, *15*, 750–757.

Osuna, E., Freund, R., & Girosi, F. (1997). An improved training algorithms for support vector machines. *Neural Networks for Signal Processing VII – Proceedings of the 1997 IEEE Workshop* (pp. 276–285). IEEE.

Platt, J. (1998). Fast training of SVMs using sequential minimal optimization. In Schölkopf et B. al. (Ed.), *Advances in kernel methods: Svm learning*. MIT Press.

Rätsch, G., Onoda, T., & Müller, K. (2001). Soft margins for AdaBoost. *Machine Learning*, *43*, 287–320.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels*. MIT Press.

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer.

Vapnik, V., & Chapelle, O. (2000). Bounds on error expectation for SVM. In Smola et A. al. (Ed.), *Advances in large margin classifiers*, 261–280. MIT Press.

Zhang, T. (2001). A leave-one-out cross validaton bound for kernel methods with application in learning. *COLT*. Springer.