SVM and Boosting: One Class

Gunnar Rätsch*, Bernhard Schölkopf†, Sebastian Mika*,
Klaus-Robert Müller*‡

*GMD FIRST, Kekuléstr. 7, 12489 Berlin, Germany
†Microsoft Research Ltd., 1 Guildhall Street, Cambridge, UK
‡University of Potsdam, Am Neuen Palais 10, 14469 Potsdam
{*raetsch, mika, klaus*}*@first.gmd.de*, *bsc@scientist.com*

November 9, 2000

Abstract

We show via an equivalence of mathematical programs that a Support Vector (SV) algorithm can be translated into an equivalent boosting-like algorithm and vice versa. We exemplify this translation procedure for a new algorithm — one-class Leveraging — starting from the one-class Support Vector Machines (1-SVM). This is a first step towards unsupervised learning in a Boosting framework. Building on so-called barrier methods known from the theory of constrained optimization, it returns a function, written as a convex combination of basis hypotheses, that characterizes whether a given test point is likely to have been generated from the distribution underlying the training data. Simulations on one-class classification problems demonstrate the usefulness of our approach.

1 Introduction

Boosting methods have successfully been applied to classification problems (Drucker et al., 1993; LeCun et al., 1995; Maclin & Opitz, 1997; Schwenk & Bengio, 1997; Bauer & Kohavi, 1999; Dietterich, 1999) and more recently also to regression estimation (Duffy & Helmbold, 2000; Rätsch et al., 2000e; Bennett et al., 2000a). Their high accuracy, ease of implementation, and wide applicability has placed them in the standard toolbox of machine learning, next to neural networks and kernel based learning methods like Support Vector Machines (SVMs) (Vapnik, 1995).

The present paper aims to point out the equivalence between the mathematical programs underlying both Boosting and SVMs, thus formalizing a correspondence which has been hinted at by researchers in the boosting community (e.g. in Schapire et al., 1997). We show that a given hypothesis set in Boosting corresponds to the choice of a particular kernel in SVMs and vice versa. We illustrate this correspondence in an interesting application: *one-class classification*, where one trains on unlabelled data, trying to assess whether a test point is likely to belong to the distribution underlying

the training data (cf. Schölkopf et al., 1999; Tax & Duin, 1999). This problem of unsupervised learning can be thought of as a simplified version of the problem of density estimation. One-class classification has so far been studied in the context of SVMs (Schölkopf et al., 1999; Tax & Duin, 1999; Campbell & Bennett, 2001), but not for Boosting.

A potential advantage in this and other boosting-like techniques is that prior knowledge can be used for the choice of the base hypotheses or weak learners. It is often easier to incorporate prior knowledge into a weak learner than into a kernel function. Specifically, in the context of one-class classification it is advantageous to work with interpretable simple base hypotheses, e.g. decision stumps or linear cuts, as this lets us extract simple and interpretable rules from the decision boundary (cf. Section 5).

The next section shows the connection between Boosting and SVMs. Using this relation, we first derive a linear programming formulation of the one-class problem in Section 3 and then develop a boosting-like technique in Section 4, building upon results of (Rätsch et al., 2000e) that connects Boosting to Barrier optimization techniques. The experimental Section 5 shows the validity of our approach. Finally a brief conclusion follows.

2 SVMs and Boosting

In the next two subsections we will start to review the basic ideas and the corresponding optimization problems of SVMs and Boosting methods. Then we will show the relation between both approaches and discuss why different norms have to be used for SVMs and Boosting. Here $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in \mathcal{X} \times \{-1, +1\}$ denotes our training sample of size N for some set $\mathcal{X} \subseteq \mathbb{R}^m$. See Appendix C, for a summary of the notation used in this paper.

2.1 Support Vector Machines

Consider an J-dimensional *feature space* $\mathcal{F}$ which is a subset of $\mathbb{R}^J$ and is spanned by a mapping Φ. In a support vector (SV) setting, any Φ corresponds to a Mercer kernel $k(\mathbf{x}, \mathbf{x}') = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}'))$ implicitly computing the dot product in $\mathcal{F}$. The goal of SVMs is to find some separating hyperplane described by a vector $\mathbf{w}$ in feature space $\mathcal{F}$:

$$f_{\mathbf{w}}(\mathbf{x}) = (\mathbf{w} \cdot \Phi(\mathbf{x}_n)).$$

Finding the hyperplane can be casted into a quadratic optimization problem:

$$\begin{array}{ll} \min\limits_{\mathbf{w} \in \mathcal{F}} & \frac{1}{2}\|\mathbf{w}\|_2^2 \\ \text{subject to} & y_n(\mathbf{w} \cdot \Phi(\mathbf{x}_n)) \geq 1 \quad n = 1, \ldots, N, \end{array} \tag{1}$$

where for simplicity we have omitted the bias/offset. One selects the hyperplane with minimal (VC) capacity (Vapnik, 1995), which is in this case can be achieved by maximizing the margin. Here, the margin $\rho = \frac{2}{\|\mathbf{w}\|_2}$ is defined as the minimum ℓ_2-distance of a training point to the separating hyperplane.

More generally, the *margin* of an example $(\mathbf{x}_n, y_n)$ is defined as the ℓ_q-distance of the pattern to a given separating hyperplane. A positive margin corresponds to a correct classification and the more positive the margin the greater the "confidence" (Schapire et al., 1997; Schapire & Singer, 1998; Vapnik, 1995) that the learned classifier is correct. The connection between the ℓ_q-margin of a pattern and the term $y_n(\mathbf{w} \cdot \Phi(\mathbf{x}_n))$ is given by

Theorem 1 (Mangasarian (1999)). *Let $\mathbf{z} \in \mathbb{R}^J$ be any point which is not on the plane $A = \{\tilde{\mathbf{z}} \mid \tilde{\mathbf{z}}^\top \boldsymbol{\alpha} = 0\}$. Then for $p \in [1, \infty]$:*

$$\frac{|\mathbf{z}^\top \boldsymbol{\alpha}|}{\|\boldsymbol{\alpha}\|_p} = \|\mathbf{z} - A\|_q, \tag{2}$$

where $\|\mathbf{z} - A\|_q$ denotes the distance $\mathbf{z}$ to the plane A measured with respect to the dual norm ℓ_q, i.e. $\frac{1}{p} + \frac{1}{q} = 1$.

Therefore, maximizing the minimal ℓ_q-distance of the training patterns to the separating hyperplane is equivalent to solving the following optimization problem (Mangasarian, 1999):

$$\begin{array}{ll} \max\limits_{\mathbf{w} \in \mathcal{F}, \rho \in \mathbb{R}} & \rho \\ \text{subject to} & y_n(\mathbf{w} \cdot \Phi(\mathbf{x}_n)) \geq \rho \quad n = 1, \ldots, N \\ & \|\mathbf{w}\|_p = 1, \end{array} \tag{3}$$

with the conditions on p and q as in the Theorem above. For $p = q = 2$ this turns out to be equivalent to solving problem (1). In this case, it has been shown (Vapnik, 1995) that the solution $\mathbf{w}$ can be expressed as a linear combination of the training patterns using some positive coefficients $\boldsymbol{\lambda}$:

$$\mathbf{w} = \sum_{n=1}^{N} \lambda_n \Phi(\mathbf{x}_n). \tag{4}$$

It turns out that although the coefficients $\boldsymbol{\lambda}$ are sparse, this does not hold for the weight vector $\mathbf{w}$ in feature space.

2.2 Boosting

Now we consider Boosting algorithms. Here, we will in particular focus on the Arc-GV algorithm (Breiman, 1997). For AdaBoost (Freund & Schapire, 1994) similar connections can be made (see discussions in Rätsch et al., 2000e). Let $\mathcal{H}$ be a class of base hypotheses $\mathcal{H} = \{h_j : \mathcal{X} \to \mathbb{R} \ : \ j = 1, \ldots, J\}$. In Boosting one seeks for a linear combination

$$f_{\mathbf{w}}(\mathbf{x}) = \textstyle\sum_{j=1}^{J} w_j h_j(\mathbf{x}), \tag{5}$$

where $\mathbf{w}$ is the vector of the hypothesis weights to be found. This is done by (i) calling a *base learner* $\mathcal{L}$ that selects a hypothesis h_t in each iteration t and then (ii) computing

the corresponding hypothesis weight $w_t \geq 0$ (cf. Freund & Schapire, 1994). The base learner $\mathcal{L}$ is given the training set and a set of weights $\mathbf{d}^t = [d_1^t, \ldots, d_N^t]$, which are updated in each iteration t – starting from a uniform distribution. $\mathcal{L}$ ideally finds the hypothesis $h \in \mathcal{H}$ that minimizes the weighted training error cf. *weighted minimization* in (cf. *weighted minimization* in Breiman, 1997). At the end of the algorithm the hypothesis weights $\mathbf{w}$ are normalized, such that they sum to 1. For details on how the weights $\mathbf{d}$ and w_t are computed in AdaBoost and Arc-GV, cf. Freund and Schapire (1994) and Breiman (1997), respectively.

Arc-GV has been shown[1] asymptotically (Breiman, 1997) to find the linear combination that solves a linear optimization problem (LP) which is commonly stated as[2]:

$$\begin{array}{ll} \max\limits_{\mathbf{w}\in\mathbb{R}_+^J, \rho\in\mathbb{R}} & \rho \\ \text{subject to} & y_n \sum_{j=1}^J w_j h_j(\mathbf{x}_n) \geq \rho \quad n = 1, \ldots, N \\ & \|\mathbf{w}\|_1 = 1, \end{array} \tag{6}$$

where $\mathbb{R}_+$ denotes the non-negative half-space of $\mathbb{R}$. The solution $\mathbf{w}$ of (6) has been found to be sparse, i.e. very few base hypotheses of $\mathcal{H}$ are combined in Eq. (5) (Bennett & Mangasarian, 1992; Chen et al., 1995; Bradley et al., 1998; Rätsch et al., 2000c). This number can be bounded by the number of patterns N (independently of the size of $\mathcal{H}$), but is usually much smaller.[3]

2.3 Connection between Boosting and SVMs

It is a common folklore statement that Boosting and SVMs are "essentially the same" except for the way they measure the margin or the way they optimize their weight vector: SVMs use the ℓ_2-norm and Boosting employs an ℓ_1-norm. One might think that this solely influences the imposed regularization. We would like to make the connection more precise and explicit and will show that they use two unique strategies to handle high or even infinite dimensional spaces. First, SVMs need to use the ℓ_2-norm to implicitly compute scalar products in feature space with the help of the kernel trick. No other norm can be expressed in terms of scalar products. Boosting, in contrast, performs the computation explicitely in feature space. This is well-known to be prohibitive if the solution $\mathbf{w}$ is not sparse, as the feature space might be very high or even infinite dimensional, depending on the size of the base hypothesis set. Therefore the ℓ_1-norm or another sparseness inducing regularization functional e.g. (e.g. Mangasarian, 1997) is mandatory. Boosting relies on the fact that there are only a few hypotheses necessary to express the solution, which Boosting tries to find during each iteration. Basically, Boosting considers only the most salient dimensions in the feature space spanned by

[1] This is also believed to be true for AdaBoost if the problem is separable, i.e. $\rho > 0$ (cf. Freund & Schapire, 1996; Rätsch et al., 2000e).

[2] We added the constraints $\mathbf{w} \geq \mathbf{0}$, often used in the context of Boosting. If one assumes complementation closeness of the hypothesis set $\mathcal{H}$, then adding the constraints $\mathbf{w} \geq \mathbf{0}$ does not change the solution.

[3] Roughly speaking, a reason for the induced sparseness is the fact that vectors far from the coordinate axes are "larger" with respect to the ℓ_1-norm than with respect to ℓ_p-norms with $p > 1$. For example, consider the vectors $(1, 0)$ and $(1/\sqrt{2}, 1/\sqrt{2})$. For the two norm, $\|(1, 0)\|_2 = \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = 1$, but for the ℓ_1-norm, $1 = \|(1, 0)\|_1 < \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = \sqrt{2}$. Note that using the ℓ_1-norm as regularizer the optimal solution is always a vertex solution (or can be expressed as such) and tends to be very sparse.

the hypotheses and can therefore be very efficient. Also on the level of the mathematical programs we can see the relation between Boosting and SVMs: (3) and (6) are clearly similar for $p = 1$. To make this explicit, note that any hypothesis set $\mathcal{H}$ implies a mapping Φ by

$$\Phi : \mathbf{x} \mapsto [h_1(\mathbf{x}), \ldots, h_J(\mathbf{x})]^\top, \tag{7}$$

and therefore also a kernel $k(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{J} h_j(\mathbf{x}) h_j(\mathbf{x}')$. Thus, any hypothesis set $\mathcal{H}$ spans a feature space $\mathcal{F}$. Furthermore, for any feature space $\mathcal{F}$ which is spanned by some mapping Φ, the corresponding hypothesis set $\mathcal{H}$ can be constructed by $h_j \equiv P_j \Phi$, where P_j denotes the projection onto the j-th dimension in feature space.

3 From One-Class SVMs to One-Class-LPs

3.1 One-Class-SVMs

The goal of the one-class SVM (1-SVM) approach of (Schölkopf et al., 1999) is to find some hyperplane $\mathbf{w} \in \mathcal{F}$ that separates the unlabeled training data from the *origin* at some threshold ρ, i.e. one estimates a function $f_{\mathbf{w}}(\mathbf{x}) = (\mathbf{w} \cdot \Phi(\mathbf{x}))$ and decides that a pattern $\mathbf{x}$ belongs to the *one class* whenever $f_{\mathbf{w}}(\mathbf{x}) \geq \rho$. To find $\mathbf{w}$ and the threshold ρ, the following quadratic program is used:

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{F}, \boldsymbol{\xi} \in \mathbb{R}^N_+, \rho \in \mathbb{R}} \quad & \tfrac{1}{2}\|\mathbf{w}\|_2^2 - \nu\rho + \tfrac{1}{N} \sum_{n=1}^{N} \xi_n \\ \text{subject to} \quad & \\ & (\mathbf{w} \cdot \Phi(\mathbf{x}_n)) \geq \rho - \xi_n, \quad n = 1, \ldots, N. \end{aligned} \tag{8}$$

The optimization problem incorporates the intuition that we would like to have a large fraction of training patterns satisfying $f_{\mathbf{w}}(\mathbf{x}) \geq \rho$, while still having a small SV-type regularization term $\|\mathbf{w}\|_2^2$ (Vapnik, 1995). The parameter $\nu \in (0, 1)$ controls the trade-off; its meaning will become clearer in the next section.

A different approach (called Support Vector Data Description – SVDD) is taken in Tax and Duin (1999), where one is not seeking for a hyperplane, but a hypersphere which contains as many as possible of the training data while keeping the radius small. When using radial basis function (RBF) kernels, this has been shown to be equivalent to the current approach (Schölkopf et al., 1999).

3.2 A Linear Programming Approach

As we intend to consider high-dimensional feature spaces $\mathcal{F}$, it can be prohibitive to have non-sparse solution vectors $\mathbf{w}$ when explicitly carrying out computations in $\mathcal{F}$. We do not want to be forced to take all J dimensions into account to decide whether a new pattern $\mathbf{x}$ belongs to the one class or not.

We therefore propose to modify the regularization term on the weight vector $\mathbf{w}$ to the ℓ_1-norm in feature space, as the ℓ_1-norm used in (8) induces sparsity in $\mathbf{w}$ (see

discussion in Section 2.2). Thus we obtain the following optimization problem:

$$\begin{array}{ll} \min\limits_{\mathbf{w}\in\mathcal{F},\boldsymbol{\xi}\in\mathbb{R}_+^N,\rho\in\mathbb{R}} & -\nu\rho + \frac{1}{N}\sum\limits_{n=1}^{N}\xi_n \\ \text{subject to} & (\mathbf{w}\cdot\Phi(\mathbf{x}_n)) \geq \rho - \xi_n, \quad n = 1,\ldots,N \\ & \|\mathbf{w}\|_1 = 1 \end{array} \tag{9}$$

where we have to fix $\|\mathbf{w}\|_1 = 1$ because the objective is – contrary to (8) – linear in $\|\mathbf{w}\|_1$ and the optimal solution would otherwise be trivial ($\mathbf{w} = \mathbf{0}$) or unbounded ($\|\mathbf{w}\|_1 \to \infty$). Note that there exists another version of (9) which instead of $\|\mathbf{w}\|_1 = 1$ uses $\rho = 1$. It can be shown to possess the same solution set, provided the optimal solution of (9) satisfies $\rho > 0$.

It is furthermore worthwhile to observe that the solutions of (8) and (9) have the ν-property (shown for (8) in Schölkopf et al., 1999):

Proposition 2. *Assume the solution of (9) satisfies $\rho \geq 0$. The following statements hold:*

(i) ν is an upper bound on the fraction of outliers (i.e. points with $f_{\mathbf{w}}(\mathbf{x}) < \rho$).

(ii) ν is a lower bound on the fraction of points which are not inside (i.e. outside or on the boundary of) the estimated region (i.e. points with $f_{\mathbf{w}}(\mathbf{x}) \leq \rho$).

The proof is analogous to the one in Schölkopf et al. (1999), Rätsch et al. (2000d). Here, we would like to emphasize that in particular in unsupervised learning it is important to have *interpretable* regularization constants, as the model selection for this case poses a difficult problem. In practice, the parameter ν can often be selected from the knowledge about the problem at hand (see also Section 5.3).

A similar algorithm has been proposed independently by Campbell and Bennett (2001), where one finds the additional term $\sum_{n=1}^{N} f_{\mathbf{w}}(\mathbf{x}_n)$ in the objective. The ν-property also holds for this case and the derivations of a boosting-like algorithm in the following sections can be adapted easily to their approach.

From Eq. (7), any mapping Φ induces some hypothesis space $\mathcal{H} = \{h_j : \mathcal{X} \to \mathbb{R} : j = 1,\ldots,J\}$. Thus we can easily replace $(\mathbf{w}\cdot\Phi(\mathbf{x}_n))$ by $\sum_{j=1}^{J} w_j h_j(\mathbf{x}_n)$ in the inequality constraints of (9). We furthermore assume that the hypothesis space is complementation closed and can enforce $w_j \geq 0$ ($j = 1,\ldots,J$) without changing the problem and get the following linear optimization problem:

$$\begin{array}{ll} \min\limits_{\mathbf{w}\in\mathbb{R}_+^J,\boldsymbol{\xi}\in\mathbb{R}_+^N,\rho\in\mathbb{R}} & -\nu\rho + \frac{1}{N}\sum\limits_{n=1}^{N}\xi_n \\ \text{with} & \sum_{j=1}^{J} w_j h_j(\mathbf{x}_n) \geq \rho - \xi_n, \qquad n = 1,\ldots,N \\ & \|\mathbf{w}\|_1 = 1, \end{array} \tag{10}$$

which looks similar to the one solved by Arc-GV (cf. Eq. (6)).

4 Leveraging approaches

In the last section we have obtained a linear programming formulation of the one-class problem which will now serve as a basis for deriving a boosting-like algorithm. Despite

the fact that this algorithm will not have the PAC-boosting property (Schapire, 1990) (and neither of the algorithms described before (except the AdaBoost algorithm itself), it works very similar to AdaBoost. However, being pedantic about not to confuse the terms, we will use "leveraging" instead.

Principally, any leveraging approach selects iteratively one hypothesis $h \in \mathcal{H}$ at a time and then updates the weight vector $\mathbf{w}$, which can be implemented in different ways. There are essentially two alternatives:

(i) Ideally, one solves the optimization problem for all hypotheses that have already been selected in last iterations, as proposed by Grove and Schuurmans (1998), Kivinen and Warmuth (1999), Bennett et al. (2000b).

Another – greedy – approach is used by the original AdaBoost/Arc-GV algorithm:

(ii) Here, one updates only the weight of the last hypothesis selected, while minimizing some exponential cost function (Breiman, 1997; Friedman et al., 1998; Mason et al., 2000; Rätsch et al., 2000c). It has been shown that this relates to coordinate descent methods, barrier optimization techniques (Rätsch et al., 2000e) and to the Bregman algorithm (Bregman, 1967; Censor & Zenios, 1997).

In the next section we start with considering examples for each of these categories for illustration, before we develop a new method.

4.1 ν-Arc and Column-Generation for AdaBoost

Note that the formulation in Eq. (10) is very similar to the problems underlying to ν-Arc (Rätsch et al., 2000d) and Column-Generation-AdaBoost (CG-AdaBoost) (Bennett et al., 2000b). The difference is simply that the label y_n appears as an additional factor on the left hand side of the constraints (cf. Eq. (6)) – which we do not have available in unsupervised learning. The first approach – ν-Arc – is based on the observation that the Arc-GV algorithm maximizes the minimum margin. Following the ideas in Rätsch et al. (2000d), one reformulates (10) as a *min-max-problem*:

$$\begin{array}{lll} \max\limits_{\mathbf{w}\in\mathbb{R}^J, \boldsymbol{\xi}\in\mathbb{R}^N_+, \rho\in\mathbb{R}} & \tilde{\rho} & \\ \text{with} & \sum_{j=1}^J w_j h_j(\mathbf{x}_n) + \xi_n - \frac{1}{\nu N}\sum\limits_{n=1}^N \xi_n \geq \tilde{\rho}, & n = 1, \ldots, N \\ & \|\mathbf{w}\|_1 = 1, & \end{array} \tag{11}$$

where we use $\tilde{\rho} = \rho - \frac{1}{\nu N}\sum_{n=1}^N$. This has the advantage that we have readily defined a *soft margin* in the sense of Rätsch et al. (2000c) on the left hand side of the inequalities. Using the optimal values for ρ – chosen such that the ν-property holds (cf. Proposition 2) – and $\xi_n = \max(0, \rho - f_{\mathbf{w}}(\mathbf{x}_n))$, this margin is then given continually to the Arc-GV algorithm instead of the real margin. From these soft margins, Arc-GV computes – as before – the weighting of the training set to obtain the next hypothesis and computes its weight by minimizing a certain error function. So, Arc-GV is

exploited as an optimization tool to solve the min-max problem (11) for large hypothesis classes. However, it is theoretically not completely clear whether this approach always converges to the optimal solution of (11).

The Column-Generation approach for AdaBoost (Bennett et al., 2000b) starts with the dual problem of a linear program (LP) that is similar to (10) and (6) and uses a technique well-known in the optimization community – Column Generation. Let us just briefly recapitulate this algorithm and apply it to our modified problem. The dual of (10) is (cf. Appendix B for the derivation)

$$\begin{array}{lll} \max\limits_{\mathbf{d}\in\mathbb{R}^J,\epsilon\in\mathbb{R}} & \epsilon & \\ \text{with} & \sum_{n=1}^N d_n h_j(\mathbf{x}_n) \leq \epsilon, & j=1,\ldots,J \\ & 0 \leq d_n \leq \frac{1}{N}, & n=1,\ldots,N \\ & \sum_{j=1}^J d_j = \nu, & \end{array} \tag{12}$$

where one has one constraint per hypothesis and box constraints on $\mathbf{d}$ in the N dimensional simplex. The algorithm iteratively selects a hypothesis[4] h_j (column j) and constructs subsets $\mathcal{H}^t = \{h_1,\ldots,h_t\} \subseteq \mathcal{H}$, $t = 1,2,\ldots$, for which it solves the problem (12) exactly, e.g. by using a simplex method. As in the original boosting algorithm in each iteration it generates a pattern weighting $\mathbf{d}^t$, that is used to find the next hypothesis. From (12) it can easily be seen which hypothesis this has to be, to make progress: the one with largest $\sum_{n=1}^N d_n h_j(\mathbf{x}_n)$ – called the *edge* in supervised learning (Breiman, 1997). Clearly, if all hypothesis have been added, the problem is solved. However, due to the smart selection rule and the sparseness of the solution, the convergence of this algorithm is much faster and at some point there is no hypothesis left which would change the current solution, i.e. for all $j = 1,\ldots,J$ holds $\sum_{n=1}^N d_n h_j(\mathbf{x}_n) \leq \epsilon$. Then the algorithm is converged to the solution of the full problem. Note that one has to be careful with the definition of the primal problem in the case of infinite hypothesis spaces; for a regression setting this has been considered in Rätsch et al. (2000a).

So far, we have presented two new approaches based on boosting techniques used for supervised learning. The first approach has the advantage that it is easy to implement, however, there is no guarantee of convergence to the optimal solution. The second approach boasts guaranteed convergence and the empirically fast performance (Bennett et al., 2000b), but it relies on the availability of a good LP-solver. In the following sections we will propose a new method based on the work in Rätsch et al. (2000e) using *barrier optimization techniques* to combine both algorithms and most of their advantages.

4.2 A One-Class Barrier Algorithm

Let us very briefly recall some basic statements about barrier optimization. For details see e.g. Rätsch et al. (2000e) and references therein. The goal of barrier optimization

[4]For ease of notation we will denote by h_t the hypothesis that is selected in the t-th iteration and by h_j the j-th hypothesis in the hypothesis set $\mathcal{H}$. We use the same notation for the corresponding weights w_t and w_j, respectively.

is to find an optimal solution of the problem

$$\min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x}) \qquad \text{with} \qquad \mathcal{S} = \{\mathbf{x} \mid c_i(\mathbf{x}) \geq 0, i = 1, \ldots, m\},$$

where f is a convex function over the non-empty convex set $\mathcal{S}$ of *feasible solutions*. This problem can be solved using a so called barrier function. The exp-barrier function (cf. Cominetti & Dussault, 1994; Doljansky & Teboulle, 1998; Mosheyev & Zibulevsky, 1999) is a particular useful choice (Rätsch et al., 2000e) for our purposes,

$$E_\beta(\mathbf{x}) = f(\mathbf{x}) + \beta \sum_{i=1}^{m} \exp\left[-\frac{c_i(\mathbf{x})}{\beta}\right], \tag{13}$$

where $\beta > 0$ denotes a penalty parameter. By finding a sequence of (unconstrained) minimizers to (13), using any sequence of β's with $\beta \to 0$, it can be shown that these minimizers converge to a global solution of the original problem (e.g. Cominetti & Dussault, 1994). The conditions for convergence are even more relaxed as shown in the following

Proposition 3 (Cominetti and Dussault (1994)). *Assume f and c_i are differentiable and convex functions. Let $\mathbf{x}^t$ be an $\delta_t \geq 0$ minimizer of*

$$E_{\beta_t}(\mathbf{x}) = f(\mathbf{x}) + \beta_t \sum_{i=1}^{m} \exp\left[-\frac{c_i(\mathbf{x})}{\beta_t}\right], \tag{14}$$

i.e. $\|\frac{\partial}{\partial \mathbf{x}} E_{\beta_t}(\mathbf{x}^t)\| \leq \delta_t$. Then, for $\delta_t, \beta_t \overset{t\to\infty}{\longrightarrow} 0$ every limit point of $\{\mathbf{x}^t\}_{t\in\mathbb{N}}$ is a solution to (13).

Let us now apply the exp–barrier technique to problem (10). We get the following barrier functional

$$E_\beta(\rho, \mathbf{w}, \boldsymbol{\xi}) = -\nu\rho + \frac{1}{N}\sum_{n=1}^{N} \xi_n + \beta \sum_{n=1}^{N} \exp\left[\frac{-\xi_n}{\beta}\right] + \exp\left[\frac{\delta_n - \xi_n}{\beta}\right], \tag{15}$$

where $\delta_n = \rho - \sum_{j=1}^{J} w_j h_j(x_n)$. We have omitted the terms corresponding to the constraints $\|\mathbf{w}\|_1 = 1$ and $\mathbf{w} \geq \mathbf{0}$ as we will maintain them outside the barrier-optimization.[5] In order to reduce the number of variables to be optimized, we find the optimal slack variables ξ_n by minimizing (15) for a given β, $\mathbf{w}$ and ρ by setting $\frac{\partial}{\partial \boldsymbol{\xi}} E_\beta = \mathbf{0}$ and solving for $\boldsymbol{\xi}$. We obtain a closed form solution:

$$\xi_n(\beta, \mathbf{w}, \rho) = \beta \log\left(1 + \exp\left[\frac{\delta_n}{\beta}\right]\right) + \beta \log N, \tag{16}$$

which – as expected[6] – satisfies $\lim_{\beta\to 0} \xi_n(\beta, \rho, \mathbf{w}) = \max(0, \delta_n)$. Note that it is difficult to find an expression for an optimal ρ which is independent of $\boldsymbol{\xi}$. As we will see later, there exists a simple way around for this problem.

[5] When computing gradients or proving the convergence of our algorithm we substitute $\mathbf{w}_j \equiv \frac{|w'_j|}{\|\mathbf{w}'\|_1}$, however in the algorithm we implement the constraints by restricting the search space.

[6] This becomes clear by (11) and Section 4.1.

Usually, in a barrier algorithm one would optimize all $J+N+1$ parameters directly until the desired precision is reached (cf. Proposition 3). But this requires to know all hypotheses in $\mathcal{H}$ in advance. Thus, we will propose a leveraging algorithm that finds one new hypothesis h_t and its weight $w_t \equiv w_{h_t}$ in each iteration. Then there is only one parameter, w_t, to be determined in each iteration. In proving the convergence one exploits that one needs only a δ-minimizer for E_β. It will turn out that the base-learner helps to estimate the ℓ_∞-norm of the gradient.

4.2.1 Selecting the hypothesis

The gradient of E_β with respect to each w_j can be computed as[7]:

$$\frac{\partial}{\partial w_j} E_\beta(\rho, \mathbf{w}, \boldsymbol{\xi}(\beta, \rho, \mathbf{w})) = \sum_{n=1}^{N} d_n (f_{\mathbf{w}}(\mathbf{x}_n) - h_j(\mathbf{x}_n)), \tag{17}$$

where $d_n = e^{(\delta_n - \xi_n)/\beta}$. Note that the maximum of $\frac{\partial}{\partial w_j} E_\beta(\rho, \mathbf{w}, \boldsymbol{\xi}(\beta, \rho, \mathbf{w}))$ is the duality gap between the primal problem (15) and its dual. It is therefore always non-positive (see Appendix B for details). To reduce E_β iteratively (for some fixed β) one may choose the hypothesis h_t such that

$$h_t = \operatorname*{argmax}_{h \in \mathcal{H}} \sum_{n=1}^{N} d_n h(\mathbf{x}_n) = \pm \operatorname*{argmax}_{h \in \mathcal{H}} \left| \frac{\partial}{\partial w_h} E_\beta \right|, \tag{18}$$

where w_h denotes the weight for the hypothesis h and we have used the complementation closeness of $\mathcal{H}$. This selection rule implements a Gauss-Southwell method (Luenberger, 1984) on the full problem (details in the Proof of Theorem 4). Furthermore it finds the hypothesis that is most responsible for the size of the duality gap.

Note that in supervised Boosting techniques this corresponds to minimizing the training error using the pattern weighting $\mathbf{d}$. In order to find the best hypothesis in the unsupervised setting the scalar-product between the weight vector and the hypothesis weight needs to be maximized.

4.2.2 Finding the weights

When the hypothesis h_t has been found, its weight w_t is computed. Essentially, this works as in AdaBoost:

$$\mathbf{w}^t = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{S}^t(\mathbf{w}^{t-1})} E_\beta(\rho, \boldsymbol{\xi}(\beta, \rho, \mathbf{w}), \mathbf{w}), \tag{19}$$

where one uses the optimal values of $\boldsymbol{\xi}$ as given above. The space $\mathcal{S}^t(\mathbf{w}^{t-1})$ is chosen such that the constraints $\|\mathbf{w}\|_1 = 1$ and $\mathbf{w} \geq \mathbf{0}$ are fulfilled, e.g.

$$\mathcal{S}^t(\mathbf{w}^{t-1}) = \{\lambda \mathbf{w}^{t-1} + (1-\lambda)\mathbf{e}^t \mid \lambda \in [0,1]\},$$

[7] For details see Lemma 5 and proof of Theorem 4.

where $\mathbf{e}_t$ is the t-th unit vector. Here, $\mathcal{S}^t(\mathbf{w}^{t-1})$ is a one-dimensional subset of the t-dimensional probability simplex $\mathcal{S}^t$, which is defined such that one can freely change the coefficient of the current hypothesis but takes care of the constraints $\|\mathbf{w}\|_1 = 1$ and $\mathbf{w} \geq \mathbf{0}$. One may also chose $\mathcal{S}^t(\mathbf{w}^{t-1})$ larger (e.g. $\mathcal{S}^t(\mathbf{w}^{t-1}) = \mathcal{S}^t$) if the minimization in (19) can be implemented efficiently. This may improve the convergence speed in practice. However, in our analysis we only consider the simpler case.

4.2.3 Convergence

The complete algorithm is summarized in Algorithm 1 as pseudo-code. The following theorem shows the convergence of the proposed algorithm to the solution of our one-class-LP:

Theorem 4. *Suppose $\mathcal{H}$ is a complementation closed, bounded and finite hypothesis set and the base learner $\mathcal{L}_{\mathcal{H}}$, when called by the algorithm in Algorithm 1, returns a hypothesis that fulfills* (18)*. Let* `next_beta` *be a function such that $\lim_{r\to\infty} \beta_r = 0$ and $\beta_r > 0$ $(r = 1, 2, \ldots)$. Then the output of the algorithm will asymptotically converge to the optimal solution of* (10)*.*

Sketch of Proof: The idea of the proof is as follows: We first show that the base learner selects the hypothesis that corresponds to the weight with largest gradient. By the convergence properties of coordinate descent methods (in particular the Gauss-Southwell-algorithm) we can conclude that the gradient of E_β with respect to any hypothesis-weight decreases for a fixed β. Iff this gradient becomes small enough, β is decreased and iteratively we will get $\beta \to 0$. Then we can apply Proposition 3 for the limiting case and have shown the desired convergence.

Special care needs to be taken about the additional constraints $\|\mathbf{w}\|_1 = 1$ and $\mathbf{w} \geq \mathbf{0}$, which we implement as $\mathbf{w} = \frac{|\mathbf{w}'|}{\|\mathbf{w}'\|_1}$, where $|w'|$ is the elementwise absolute value. Let $f_{\mathbf{w}'} = \sum_{j=1}^{J} w'_j h_j$ and we get the following objective

$$\begin{aligned} E_\beta(\rho, \mathbf{w}', \boldsymbol{\xi}) \quad &= \quad -\nu\rho + \frac{1}{N}\sum_{n=1}^{N} \xi_n + \beta \sum_{n=1}^{N} \exp\left[\frac{-\xi_n}{\beta}\right] + \\ &\quad + \exp\left[\frac{\rho - \xi_n - \|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\beta}\right]. \end{aligned}$$

The derivatives $\frac{\partial E_\beta}{\partial \mathbf{w}'}$ always using the optimal $\boldsymbol{\xi}(\beta, \rho, |\mathbf{w}'|/\|\mathbf{w}'\|_1)$ can easily be computed:

Lemma 5.

$$\frac{\partial E_\beta(\rho, \mathbf{w}', \boldsymbol{\xi}(\beta, \rho, |\mathbf{w}'|/\|\mathbf{w}'\|_1))}{\partial w'_j} \quad = \quad \frac{\operatorname{sign}(w'_j)}{\|\mathbf{w}'\|_1} \sum_{n=1}^{N} \left(\frac{f_{\mathbf{w}'}(\mathbf{x}_n)}{\|\mathbf{w}'\|_1} - h_j(\mathbf{x}_n) \right) d_n,$$

where $d_n = \exp\left[\frac{\rho - \xi_n(\beta, \rho, |\mathbf{w}'|/\|\mathbf{w}'\|_1) - \|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\beta}\right]$.

Algorithm 1 The 1-Class-Leveraging algorithm

argument: Sample $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, No. of iterations T, $\nu \in (0,1)$
returns: Convex combination from $\mathcal{H}$.
function 1-Class-Leverage(X, T, ν)
  Set $\beta > 0$, $\rho^0 = 0$ and $\mathbf{w}^0 = []$
  Set $d_n^1 = \nu/N$, $n = 1, \ldots, N$
  for $t = 1, \ldots, T$
    $h_t = \mathcal{L}(X, \mathbf{d}^t)$
$*$ **if** $\sum_{n=1}^{N} d_n^t (h_t(\mathbf{x}_n) - f_{\mathbf{w}^{t-1}}(\mathbf{x}_n)) \leq \beta$ **do** $\beta =$ next_beta(β) **endif**
    $\mathbf{w}^t = \operatorname{argmin}_{\mathbf{w} \in \mathcal{S}^t(\mathbf{w}^{t-1})} E_\beta(\mathbf{w}, \rho^{t-1})$
    $\rho^t = \operatorname{argmin}_{\rho \in \mathbb{R}} E_\beta(\mathbf{w}^t, \rho)$
    $d_n^{t+1} = \exp\left[\frac{\rho^t - \xi_n(\beta, \rho^t, \mathbf{w}^t) - f_{\mathbf{w}^t}(\mathbf{x}_n)}{\beta}\right]$, $n = 1, \ldots, N$
  endfor
  return $f = \sum_{t=1}^{T} w_t h_t$
end
function $E_\beta(\mathbf{w}, \rho)$
  Set $\boldsymbol{\xi} = \boldsymbol{\xi}(\beta, \rho, \mathbf{w})$ by (16)
  return $-\nu\rho + \sum_{n=1}^{N} \frac{\xi_n}{N} + \beta \exp\left[-\frac{\xi_n}{\beta}\right] + \beta \exp\left[\frac{\rho - \xi_n - f_{\mathbf{w}}(\mathbf{x}_n)}{\beta}\right]$
end

For the proof see Appendix A. By duality arguments (cf. Appendix B), the edge of the combined hypothesis is always greater or equal than the best single hypothesis. By complementation closeness of $\mathcal{H}$, for such a hypothesis there exists a hypothesis that has the same edge, but with opposite sign. Selecting the hypothesis according to (18), is therefore the same as choosing the one whose weight has the maximal gradient (which is always non-positive).

This shows that the algorithm for a fixed β implements a coordinate descent method in the variables $\mathbf{w}'$ and ρ on the objective $E_\beta(\rho, \mathbf{w}', \boldsymbol{\xi}(\beta, \rho, |\mathbf{w}'|/\|\mathbf{w}'\|_1))$. In each iteration the first coordinate w_j with maximal gradient and the the coordinate ρ is optimized. By standard convergence properties of those algorithms cf. e.g. (cf. e.g. Luenberger, 1984) the norm of the gradient of E_β with respect to $\mathbf{w}'$ and ρ will vanish asymptotically and will therefore be smaller than β after a finite number of iterations. In a sense the base-learner helps to estimate the ℓ_∞-norm of the remaining gradient and the size of the duality gap.

By the condition $*$ in the algorithm, β is (asymptotically) decreased iff the gradient gets small enough. We have therefore constructed sequences of $\mathbf{w}'$ and ρ which fulfill Proposition 3 and have eventually proven the convergence. □

4.3 Examples for Base Learner

In this section we briefly look at some base learners that could be used together with the algorithm above. Which of them should be used for solving a certain problem depends of course on the problem at hand. It is important to note that our approach is general enough to use specialized hypothesis classes designed for particular problems, such that either the performance or the interpretability is as good as desired.

We first consider base learners that itself are linear combinations of some fixed functions k_q:

$$h_{\boldsymbol{\alpha}}(\mathbf{x}) = \sum_{q=1}^{Q} \alpha_q k_q(\mathbf{x})$$

The functions k_q could e.g. be kernel functions, i.e. $k_q(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}_q)$, centered around the training patterns.[8] To find the optimal hypothesis for some weighting $\mathbf{d}$, the term $\sum_{n=1}^{N} d_n h_{\boldsymbol{\alpha}}(\mathbf{x}_n)$ needs to be maximized (cf. (18)). As $\mathcal{H}$ needs to be bounded, $\boldsymbol{\alpha}$ and $k_q(\cdot)$, $q = 1, \ldots, Q$, need to be bounded, i.e. one may maximize

$$\begin{aligned} E(\boldsymbol{\alpha}) \quad &= \quad \sum_{n=1}^{N} d_n h_{\boldsymbol{\alpha}}(\mathbf{x}_n) \\ \text{s.t.} \qquad & \quad \|\boldsymbol{\alpha}\|_p = \text{const.}; \end{aligned}$$

The next two paragraphs consider special cases for which particular simple solutions exist.

4.3.1 Sparse Combinations ($p = 1$)

For this case the minimization of $E(\boldsymbol{\alpha})$ has a very simple solution. It has been shown (Rätsch et al., 2000a) that

$$\max_{\boldsymbol{\alpha}} \sum_{n=1}^{N} d_n h_{\boldsymbol{\alpha}}(\mathbf{x}_n) \qquad \text{s.t.} \quad \|\boldsymbol{\alpha}\|_1 = 1$$

has the solution $\alpha_q = \begin{cases} 1 & g_q = \max_{q'} g_{q'} \\ 0 & \text{otherwise} \end{cases}$, where $g_q = \sum_{n=1}^{N} d_n k_q(\mathbf{x}_n)$ and assuming the maximum $g_q = \max_{q'} g_{q'}$ is unique. This implies that it is in this case eventually not useful to optimize a linear combination of functions k_q, $q = 1, \ldots, Q$, as only one of them is selected anyway. Instead, one would use the much simpler $\mathcal{H} = \{k_q \mid q = 1, \ldots, Q\}$ as hypothesis set and obtains the same solution.

[8] In Rätsch et al. (2000a) the concept of *active kernels* has been proposed also known as *moving centers* in (also known as *moving centers* in T.Poggio & Girosi, 1990), where one is even optimizing the centers around which the kernel functions are computed. However, then the hypothesis space becomes infinite dimensional and one needs a more deeper treatment of primal/dual problems and the convergence of the corresponding optimization algorithms (cf. Rätsch et al. (2000a)). Here, we restrict us to the finite hypothesis case.

4.3.2 Least Squares ($p = 2$)

Under the assumption that $\|h\|_2 = \text{const.}$ for all $h \in \mathcal{H}$, it has been pointed out (Rätsch et al., 2000e; Bennett et al., 2000b) that (18) can be solved by minimizing the least-square error between the weight vector and the hypothesis output. However, usually this assumption does not hold. Then one may solve

$$\boldsymbol{\alpha} \quad = \quad \operatorname*{argmin}_{\boldsymbol{\alpha}} \sum_{n=1}^{N} (d_n - h_{\boldsymbol{\alpha}}(\mathbf{x}_n))^2 - h_{\boldsymbol{\alpha}}(\mathbf{x}_n)^2 + C\|\boldsymbol{\alpha}\|_2^2,$$

where we use $\|\boldsymbol{\alpha}\|_2^2$ as a regularizer that effectively bounds the ℓ_2-norm of $\boldsymbol{\alpha}$ cf. (cf. Smola, 1998). The problem has a simple solution: $\alpha_q = \frac{1}{C}\sum_{n=1}^{N} d_n k_q(\mathbf{x}_n)$. This case is in particular interesting when using neural networks which linearly combine several units for the output layer. The structure of the network may e.g. implement some prior knowledge about the problem at hand.

4.3.3 Decision Stumps

As an even simpler and therefore highly interpretable hypotheses class, we consider decision stumps as frequently used in the Boosting community, however, in a slightly different form:

$$\mathcal{H}^s = \{h_{c,i}(\mathbf{x}) = \tanh((x_n - c)s) \mid c \in \{\text{set of possible cuts}\}, i = 1, \ldots, \dim(X)\}$$

where s is the *slope parameter*. Each hypothesis h is a *soft decision stump* considering coordinate i for splitting at position c. Essentially, linear combinations of those hypotheses can describe **boxes** in input space (or subspaces, cut in directions of the coordinate axes), which can be interpreted easily.

5 Experiments

In the following experiments we will illustrate that our algorithms do what they should do – detecting anomalities and outliers. As we would like to conduct experiments on which we can evaluate the result by an objective measurement, e.g. how many outliers have been detected and how many normal patterns have been classified as outliers, we need to have labels that are used for evaluation purposes only.

5.1 Toy data

We begin with a toy experiment to show some basic properties of the algorithms. The data (cf. Figure 1) have been generated from two uniform distributions with differently sized supports. This leads to a large density of patterns in the center, and to a lower density in the periphery. In the toy model, the latter are considered as outliers.

We used decision stumps of the form described in Section 4.3 with $s = 4$. Figure 1 shows the result for the barrier algorithm (cf. Algorithm 1) and the CG-algorithm described in Section 4.1. It illustrates how the parameter ν (from top ($\nu = 0.1$) to

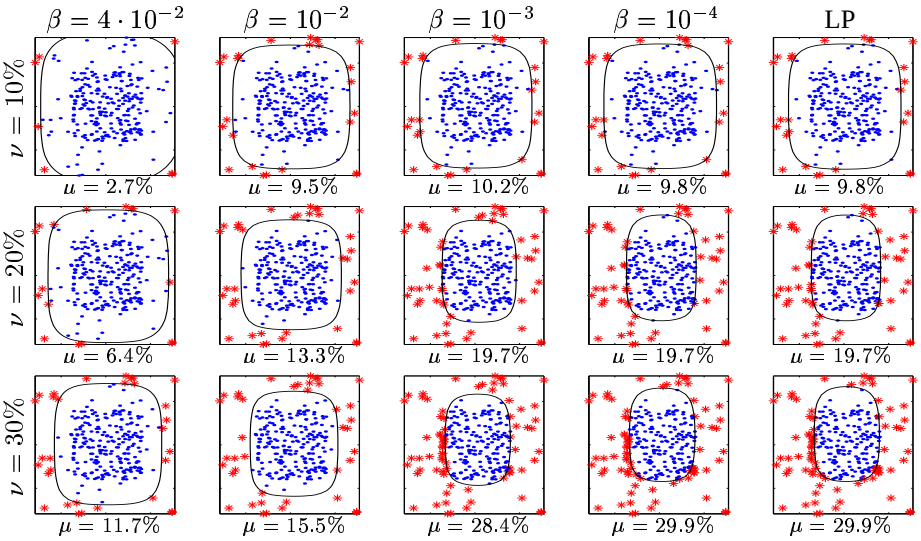


Figure 1: The One-Class approach using tanh-decision stumps for finding outliers of this toy problem for different values of ν (10% (top line), 20% (middle line) and 30% (bottom line)) for different values of β in our one-class barrier algorithm (first 4 columns) and for the one-class LP (right most column). All points are initially unlabeled; points outside the region estimated by the algorithm are marked as asterisks. The fraction μ of outliers found by the algorithms is given below the figure.

bottom ($\nu = 0.3$)) controls the fraction of outliers (marked as asterisks). Given a value of ν, the algorithm shrinks the region to be small, yet, to have no more outliers than a fraction ν. The resulting decision boundary is a superposition of four (soft) decision stumps, yielding a readily interpretable result: it can be translated into simple rules characterizing normality and novelty, outlier or exceptional event.

Furthermore, we show snapshots of the barrier-algorithm at certain levels of β at $\beta = 4 \cdot 10^{-2}$, 10^{-2}, 10^{-3} and 10^{-4}. It shows that for $\beta \to 0$ the output of our barrier-algorithm converges to the output of a linear problem solver (we used CPLEX) on the right-most. Note, that in some cases the result of the barrier-algorithm for some $\beta > 0$ describes the data intuitively better – e.g. the case $\beta = 10^{-3}$ and $\nu = 0.3$ looks a bit more appropriate than the LP solution for $\nu = 0.2$. We conjecture that this is due to the implicit regularization properties of the barrier approach.[9]

5.2 Experiment on USPS database

In a third experiment, we utilized the USPS database of handwritten characters. As base hypothesis set we use RBF kernel functions $k(\mathbf{x}, \mathbf{x}_n) = \exp\left(\frac{\|\mathbf{x}-\mathbf{x}_n\|_2}{0.3 \cdot 256}\right)$ centered around the training patterns, which has been shown useful in several independent studies cf. e.g. (cf. e.g. Schölkopf, 1997). Figure 2 shows a plot of the outputs on training

[9]By the dual of the barrier optimization problem (15) given in Appendix B, one sees that for $\beta > 0$ one is forcing the distribution vector $\mathbf{d}$ to have a small relative entropy to the uniform distribution (cf. Rätsch et al. (2000e) for some more details). Note that, differently from the CG-approach (which is favoring sparse patterns weights $\mathbf{d}$), the barrier approach tries to avoid this sparseness by using an exponential form (cf. (21)).

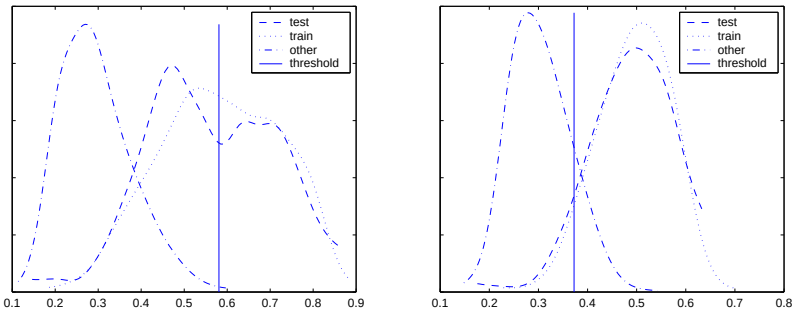


Figure 2: Experiments on the US postal service OCR data set. Recognizer for digit 0; output histogram for the exemplars of 0 in the training/test set, and on test exemplars of other digits. The x-axis gives the output values, i.e. the argument of the sign in the decision function. For $\nu = 50\%$ *(left)*, we get 49.9% outliers (consistent with Proposition 2), 43.7% true positive test examples, and only *one* false positive from the "other" class, i.e. the other digits. For $\nu = 5\%$ *(right)*, we get 4.8% outliers. In that case, the true positive rate is improved to 92.2%, while the false positive rate increases to 13.7%. The threshold ρ is marked in the graphs. The plots show a Parzen windows density estimate of the output histograms. In reality, many examples sit exactly at the threshold value.

and test sets of the US postal service database of handwritten digits. The database contains 9298 digit images of size $16 \times 16 = 256$; the last 2007 constitute the test set. We fed our algorithm with the training instances of digit 0 only. Testing was done on both digit 0 and on all other digits. As shown in Figure 2, $\nu = 50\%$ leads to *one* false positive (i.e. even though the learning machine has not seen any non-0-s during training, it correctly identifies almost all non-0-s as such), while still recognizing 43.7% of the digits 0 on the test set. Higher recognition rates can be achieved using a smaller ν: for $\nu = 5\%$, we get 92.2% correct recognition of digits 0 on the test set, with still a fairly moderate false positive rate of 13.7%. Similar experiments have been done in Schölkopf et al. (1999) for the 1-SVM, with similar results.

5.3 Model Selection

A serious problem in the one-class approach is how to select the model. Note that commonly used model-selection algorithms, such as cross-validation, are not easily applicable to the one-class problem. First one has to find an appropriate base-learner. This might be the smaller problem, as one can often derive a specialized algorithm for certain problems (e.g. RBF kernels in OCR). Furthermore one has to find the optimal parameter ν. Due to the meaning of ν one may already be able to infer the optimal ν for the problem at hand: set ν to the estimated fraction of outliers in the training data. However, in some cases even this fraction might be unknown. For this case we propose a simple heuristics that may help to find an appropriate ν.

The idea is as follows: For any ν, the one-class-classifier will find some decision irrespectively of the quality of separation between the one-class and the other patterns. Intuitively, a good separation is achieved, if the separation is very clear, i.e. the distance between the classes is large. We propose to use exactly this intuition as a criterion for model-selection. For a given ν one measures the average output of the classifier for

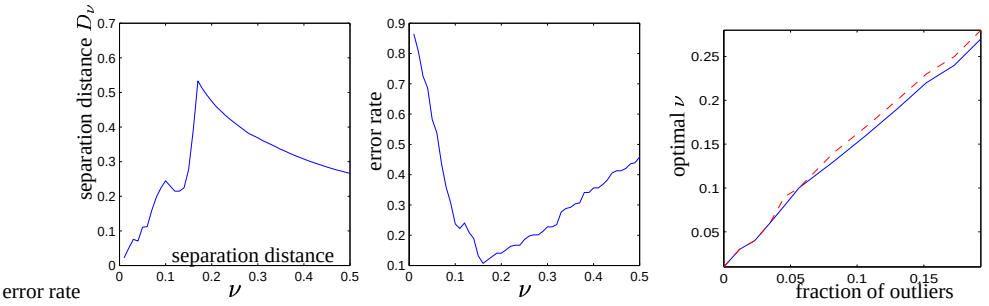


Figure 3: Digit "1" against the others: On a data set with 1005 "1"-digits and 120 randomly sampled other digits, the one-class-LP classifier was applied. The separation distance D_ν (left) and the error rate of misclassification on a labeled test set (right) for different values of ν is shown. The maximum in the separation distance coincides with the minimum in the test-error.

Figure 4: Data set with 1005 "1"-digits and different fractions (x-axis) of randomly sampled other digits: The dashed graph shows the ν selected by our maximum-separation criterion. The solid graph shows the ν that minimizes the test-error.

both classes and selects the ν which separated both classes best.

To illustrate this idea, we conduct an experiment on the USPS data set (cf. last section), where we separate digit "1" against the rest. For training we use the 1-digits and additionally include different fractions $\mu = 1\%, \ldots, 25\%$ of the other digits. For each of these data sets, we compute the one-class-LP (9) for different values of ν.[10] For each ν we compute the separation distances: $D_\nu = \frac{1}{N_-} \sum_{f_{\mathbf{w}}(\mathbf{x})<\rho} f_{\mathbf{w}}(\mathbf{x}) - \frac{1}{N_+} \sum_{f_{\mathbf{w}}(\mathbf{x})\geq\rho} f_{\mathbf{w}}(\mathbf{x})$ (where N_+ and N_- are the respective number of patterns in each of the classes) and chose the $\nu \in (0,1)$ with largest separation distance. In our example the maximum is very distinct (cf. Figure 3 left) and coincides very well with the minimum in the test error (cf. Figure 3 right and Figure 4). In some cases characterized by a very small gap between in- and outliers, we observed that the maximum D_ν is erroneously achieved for $\nu \to 1$ or $\nu \to 0$. This is to be expected as our model selection heuristic assumes a clear separation between in- and outliers.

6 Conclusion

We have studied the correspondence between SVMs and boosting-like methods. Essentially, they both work in very high-dimensional feature spaces. They differ, however, in how they deal with the algorithmic problems that this can cause. One can think of boosting as a SV approach in a high dimensional feature space spanned by the basis hypothesis. The problem becomes tractable since effectively a 1-norm regularizer is used. This induces sparsity, hence we never really work in the full space, but always in a small subspace. Vice versa, one can think of the SVM as a "boosting" approach in a high-dimensional space. In SVMs, we use the kernel trick and therefore never

[10] Good optimization packages (e.g. CPLEX) have the very useful feature that one allows one to change coefficients in the objective function (in our case ν) and to obtain an updated solution with low computational effort.

explicitly work in the feature space. Therefore, SVMs get away without having to use 1-norm regularizers; indeed, they *could not* use them, as the kernel only allows computation of the ℓ_2-norm in feature space. Both methods lead to sparse solutions, either in sample (coefficient) space (SVMs), or in feature space (boosting), and both methods are adapted to algorithmically exploit the form of sparsity they produce.

Besides providing insight, this correspondence has concrete practical benefits for designing new algorithms. We have employed it to devise a leveraging algorithm for novelty detection. The new algorithm combines the ideas and benefits of ν-Arc and the column-generation algorithm for Boosting. It converges asymptotically to the solution of a linear program similar to the 1-SVM program.

In experiments we have shown the promise of a new research direction in boosting: unsupervised learning. The focus of our contribution is to be seen on the theoretical and conceptual side. On the practical side this paper has conducted experiments on both: toy data and real-world OCR data, to demonstrate the proof of concept. Other interesting real-world one-class applications are e.g. the challenging splice-site detection problem on DNA (cf. Salzberg (1997), Rätsch, Jagota, and Müller (2000b)).

Future theoretical research will be dedicated to incorporate prior knowledge to obtain one-class algorithms that are eventually faster, better, more general, and easier to understand and interpret. In particular the extension of our algorithms to infinite hypothesis classes and the derivation of more theoretically motivated model-selection methods for the one-class classification problem are challenging.

Acknowledgments We thank for valuable discussions with Manfred Warmuth, Alex Smola, Bob Williamson, and Ayhan Demiriz. This work was partially funded by DFG under contract JA 379/91, JA 379/71, MU 987/1-1 and by EU in the NeuroColt2 project. Furthermore, GR would like to thank CRIEPI, ANU and UC Santa Cruz for warm hospitality.

A Proof of Lemma 5

Proof: It holds

$$\frac{1}{N}\exp\left[\frac{\xi_n(\beta,\rho,\mathbf{w}')}{\beta}\right] = 1+\exp\left[\frac{\rho-\|\mathbf{w}'\|_1^{-1}f_{\mathbf{w}'}(\mathbf{x}_n)}{\beta}\right], \quad n=1,\ldots,N. \tag{20}$$

We have

$$\begin{aligned}
&\frac{\partial E_\beta(\rho,\mathbf{w}',\boldsymbol{\xi}(\beta,\rho,\mathbf{w}'))}{\partial w_j'} = \\
&\frac{1}{N}\sum_{n=1}^N \frac{\partial\xi_n(\beta,\rho,\mathbf{w}')}{\partial w_j'} - \sum_{n=1}^N \frac{\partial\xi_n(\beta,\rho,\mathbf{w}')}{\partial w_j'}\exp\left[\frac{-\xi_n(\beta,\rho,\mathbf{w}')}{\beta}\right] \\
&+\left(\frac{\partial\xi_n(\beta,\rho,\mathbf{w}')}{\partial w_j'} + \frac{\partial\|\mathbf{w}'\|_1^{-1}f_{\mathbf{w}'}(\mathbf{x}_n)}{\partial w_j'}\right)\exp\left[\frac{\rho-\xi_n(\beta,\rho,\mathbf{w}')-\|\mathbf{w}'\|_1^{-1}f_{\mathbf{w}'}(\mathbf{x}_n)}{\beta}\right]
\end{aligned}$$

and by (20) we get

$$\begin{aligned}\frac{\partial \xi_n(\beta,\mathbf{w}',\rho)}{\partial w'_j} &= -\frac{\frac{\partial \|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\partial w'_j}\exp\left[\frac{\rho-\|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\beta}\right]}{1+\exp\left[\frac{\rho-\|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\beta}\right]} \\ &= -N\frac{\partial \|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\partial w'_j}\exp\left[\frac{\rho-\|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)-\xi_n}{\beta}\right].\end{aligned}$$

Plugging in yields:

$$\begin{aligned}&\frac{\partial E_\beta(\rho,\mathbf{w}',\boldsymbol{\xi}(\beta,\rho,\mathbf{w}'))}{\partial w'_j} \\ =& \sum_{n=1}^N \frac{\partial \xi_n(\beta,\rho,\mathbf{w}')}{\partial w'_j}\left(\frac{1}{N}-\exp\left[\frac{-\xi_n(\beta,\rho,\mathbf{w}')}{\beta}\right]-d_n\right)-\sum_{n=1}^N \frac{\partial \|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\partial w'_j}d_n, \\ =& -\sum_{n=1}^N \frac{\partial \|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\partial w'_j}d_n\left(2-N\exp\left[\frac{-\xi_n(\beta,\rho,\mathbf{w}')}{\beta}\right]-Nd_n\right) \\ =& -\sum_{n=1}^N \frac{\partial \|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\partial w'_j}d_n = \frac{\operatorname{sign}(w'_j)}{\|\mathbf{w}'\|_1}\sum_{n=1}^N(\|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)-h_j(\mathbf{x}_n))d_n,\end{aligned}$$

where we used (20) in the second last line, $\frac{\partial \|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\partial w'_j} = \operatorname{sign}(w_j)\frac{h_j(\mathbf{x}_n)-\|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\|\mathbf{w}'\|_1}$ in the last line and have set $d_n = \exp\left[\frac{\rho-\xi_n(\beta,\rho,\mathbf{w}')-\|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\beta}\right]$. □

B Primal/Dual problem and the Duality Gap

Let us start with the following problem (set $\beta \to 0$ for the corresponding LP):

$$\begin{aligned}\min \quad & \epsilon+\beta\sum_{n=1}^N d_n(\log d_n-1)+q_n(\log q_n-1) \\ \text{s.t.} \quad & d^\top\mathbf{1}=\nu,\ d_n+q_n=\frac{1}{N} \\ & A^\top\mathbf{d}\le\epsilon\mathbf{1},\end{aligned}$$

where we use $A_{nj}=h_j(\mathbf{x}_n)$ for simplicity. The Lagrangian for this problem is

$$\begin{aligned}L(\epsilon,\mathbf{d},\mathbf{w},\rho) &= \epsilon+\beta\sum_{n=1}^N d_n(\log d_n-1)+q_n(\log q_n-1)+ \\ &\quad +\rho(\nu-\mathbf{d}^\top\mathbf{1})+\boldsymbol{\xi}^\top(\mathbf{d}+\mathbf{q}-\frac{1}{N}\mathbf{1})+\mathbf{w}^\top(A^\top\mathbf{d}-\epsilon\mathbf{1})\end{aligned}$$

s.t. $\mathbf{w}\ge\mathbf{0}$. Setting the derivatives with respect to the primal variables to zero, yields $\|\mathbf{w}\|_1=1$ and two non-linear constraint-sets

$$d_n=\exp\left(\frac{\rho-\xi_n-A_n\mathbf{w}}{\beta}\right) \quad \text{and} \quad q_n=\exp\left(\frac{-\xi_n}{\beta}\right). \tag{21}$$

Plugging in $A_n\mathbf{w} = \rho - \beta \log d_n - \xi_n$ and $\xi_n = -\beta \log q_n$:

$$\begin{aligned} L(\epsilon, \mathbf{d}, \mathbf{w}, \rho, \boldsymbol{\xi}) &= \beta \sum_{n=1}^{N} d_n(\log d_n - 1) + q_n(\log q_n - 1) + \rho(\nu - \mathbf{d}^\top \mathbf{1}) - \\ &\quad -\mathbf{d}^\top(\beta \log \mathbf{d} + \boldsymbol{\xi} - \rho \mathbf{1}) + \boldsymbol{\xi}^\top(\mathbf{d} + \mathbf{q} - \frac{1}{N}\mathbf{1}) \\ &= \nu\rho - \frac{1}{N}\boldsymbol{\xi}^\top \mathbf{1} - \beta \mathbf{d}^\top \mathbf{1} - \beta \mathbf{q}^\top \mathbf{1}, \end{aligned}$$

and we get the dual program

$$\begin{aligned} \max \quad & \nu\rho - \frac{1}{N}\boldsymbol{\xi}^\top \mathbf{1} - \beta \sum_{n=1}^{N} \exp\left[\frac{\rho - \xi_n - A_n\mathbf{w}}{\beta}\right] + \exp\left[-\frac{\xi_n}{\beta}\right] \\ \text{s.t.} \quad & \mathbf{w} \geq \mathbf{0}, \|\mathbf{w}\|_1 = 1, \end{aligned}$$

which is the same as stated in (15).

The duality gap, i.e. the difference between primal and dual objective, at optimal (and feasible) $\mathbf{d}$, $\mathbf{q}$ and $\epsilon = \max_j \sum_n d_n A_{nj}$ is given by:

$$\begin{aligned} \text{GAP} &= \beta \sum_{n=1}^{N} d_n(\log d_n - 1) + q_n(\log q_n - 1) + \epsilon - \nu\rho + \frac{1}{N}\boldsymbol{\xi}^\top \mathbf{1} + \beta \sum_{n=1}^{N} d_n + q_n \\ &= \max_{j=1,\dots,J} \sum_n d_n(A_{nj} - A_n\mathbf{w}) \geq 0. \end{aligned} \tag{22}$$

C Notation

We use the following notational conventions:

n, N	counter and number of patterns
j, J	counter and number of hypotheses, dimensionality of $\mathcal{F}$
t, T	counter and number of iterations
$\mathcal{X}$	input space
$\mathbb{R}_+$	space of non-negative real numbers
$\mathbf{x}, y$	a training pattern and the label
$\mathcal{H}, h_j$	set of base hypotheses and an element
$\mathcal{F}$	feature space
$\mathbf{w}$	hypothesis weight vector
ξ_n	the "slack-variable" for pattern $\mathbf{x}_n$
$f_\mathbf{w}$	an combined hypothesis using the weighting $\mathbf{w}$
$\mathbf{d}$	weighting on the training set
ν	the quantile parameter (determines the number of outliers)
ρ	the margin
β	barrier penalty parameter
$\|\cdot\|_p$	the ℓ_p-norm, $p = [1, \infty]$
$k(\cdot,\cdot)$	scalar product in feature space $\mathcal{F}$

References

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithm: Bagging, boosting and variants. *Machine Learning*, *36*, 105–142.

Bennett, K., Demiriz, A., & Shawe-Taylor, J. (2000a). A column generation algorithm for boosting. Unpublished manuscript, Submitted to the Special Issue of Machine Learning.

Bennett, K., Demiriz, A., & Shawe-Taylor, J. (2000b). A column generation algorithm for boosting. In Langley, P. (Ed.), *Prooceedings, 17th ICML*, pp. 65–72 San Francisco. Morgan Kaufmann.

Bennett, K., & Mangasarian, O. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, *1*, 23–34.

Bradley, P., Mangasarian, O., & Rosen, J. (1998). Parsimonious least norm approximation. *Computational Optimization and Applications*, *11*(1), 5–21.

Bregman, L. (1967). The relaxation method for finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, *7*, 200–127.

Breiman, L. (1997). Prediction games and arcing algorithms. Technical report 504, Statistics Department, University of California.

Campbell, C., & Bennett, K. (2001). A linear programming approach to novelty detection. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*. MIT-Press. to appear.

Censor, Y., & Zenios, S. (1997). *Parallel Optimization: Theory, Algorithms and Application*. Numerical Mathematics and Scientific Computation. Oxford University Press.

Chen, S., Donoho, D., & Saunders, M. (1995). Atomic decomposition by basis pursuit. Tech. rep. 479, Department of Statistics, Stanford University.

Cominetti, R., & Dussault, J.-P. (1994). A stable exponential penalty algorithm with superlinear convergence. *J.O.T.A.*, *83*(2).

Dietterich, T. (1999). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, *40*(2).

Doljansky, M., & Teboulle, M. (1998). An interior proximal algorithm and the exponential multiplier method for semidefinite programming. *SIAM J. Optim.*, *9*(1), 1–13.

Drucker, H., Schapire, R., & Simard, P. (1993). Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, *7*, 705 – 719.

Duffy, N., & Helmbold, D. (2000). Leveraging for regression. In *Proc. COLT'00*, pp. 208–219 Stanford. Morgan Kaufmann.

Freund, Y., & Schapire, R. (1996). Game theory, on-line prediction and boosting. In *Proc. COLT*. Morgan Kaufman.

Freund, Y., & Schapire, R. (1994). A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT: European Conference on Computational Learning Theory*. LNCS.

Friedman, J., Hastie, T., & Tibshirani, R. (1998). Additive logistic regression: a statistical view of boosting. Tech. rep., Department of Statistics, Sequoia Hall, Stanford Univerity.

Grove, A., & Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artifical Intelligence*.

Hayton, P., Schölkopf, B., Tarassenko, L., & Anuzis, P. (2001). Support vector novelity detection applied to jet engine vibration spectra. In *Proc. NIPS'2000*. to appear.

Kivinen, J., & Warmuth, M. (1999). Boosting as entropy projection. In *Proc. COLT'99*.

LeCun, Y., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Müller, U. A., Säckinger, E., Simard, P., & Vapnik, V. (1995). Comparison of learning algorithms for handwritten digit recognition. In Fogelman-Soulié, F., & Gallinari, P. (Eds.), *Proc. ICANN'95*, Vol. II, pp. 53–60 Nanterre, France.

Luenberger, D. (1984). *Linear and Nonlinear Programming* (second edition). Addison-Wesley Publishing Co., Reading. Reprinted with corrections in May, 1989.

Maclin, R., & Opitz, D. (1997). An empirical evaluation of bagging and boosting. In *Proc. of AAAI*.

Mangasarian, O. (1997). Mathematical programming in data mining. *Data Mining and Knowledge Discovery*, *42*(1), 183–201.

Mangasarian, O. (1999). Arbitrary-norm separating plane. *Operation Research Letters*, *24*(1), 15–23.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Functional gradient techniques for combining hypotheses. In Smola, A., Bartlett, P., Schölkopf, B., & Schuurmans, D. (Eds.), *Advances in Large Margin Classifiers*, pp. 221–247. MIT Press, Cambridge, MA.

Mosheyev, L., & Zibulevsky, M. (1999). Penalty/barrier multiplier algorithm for semidefinite programming. *Optimization Methods and Software*. In press.

Rätsch, G., Demiriz, A., & Bennett, K. (2000a). Sparse regression ensembles in infinite and finite hypothesis spaces. *Special Issue of Machine Learning*. submitted.

Rätsch, G., Jagota, A., & Müller, K.-R. (2000b). Splice-site recognition with support vector machines. in preparation.

Rätsch, G., Onoda, T., & Müller, K.-R. (2000c). Soft margins for AdaBoost. *Machine Learning*. also NeuroCOLT Technical Report NC-TR-1998-021.

Rätsch, G., Schölkopf, B., Smola, A., Mika, S., Onoda, T., & Müller, K.-R. (2000d). Robust ensemble learning. In Smola, A., Bartlett, P., Schölkopf, B., & Schuurmans, D. (Eds.), *Advances in Large Margin Classifiers*, pp. 207–219. MIT Press, Cambridge, MA.

Rätsch, G., Warmuth, M., Mika, S., Onoda, T., Lemm, S., & Müller, K.-R. (2000e). Barrier boosting. In *Proc. COLT*, pp. 170–179 Stanford. Morgan Kaufmann.

Salzberg, S. (1997). A method for identifying splice sites and translational start sites in eukaryotic mRNA. *Computational Applied Bioscience*, *13*(4), 365–376.

Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, *5*(2), 197–227.

Schapire, R., Freund, Y., Bartlett, P., & Lee, W. (1997). Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pp. 322–330. Morgan Kaufmann.

Schapire, R., & Singer, Y. (1998). Improved boosting algorithms using confidence-rated predictions. In *Proc. COLT'98*, pp. 80–91.

Schölkopf, B. (1997). *Support vector learning*. Oldenbourg Verlag.

Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., & Williamson, R. (1999). Estimating the support of a high-dimensional distribution. TR 87, Microsoft Research, Redmond, WA. To appear in *Neural Computation*.

Schwenk, H., & Bengio, Y. (1997). AdaBoosting neural networks. In Gerstner, W., Germond, A., Hasler, M., & Nicoud, J.-D. (Eds.), *Proc. ICANN'97*, Vol. 1327 of *LNCS*, pp. 967–972. Springer.

Smola, A. J. (1998). *Learning with Kernels*. Ph.D. thesis, Technische Universität Berlin.

Tax, D., & Duin, R. (1999). Data domain description by support vectors. In Verleysen, M. (Ed.), *Proc. ESANN*, pp. 251–256 Brussels. D. Facto Press.

T.Poggio, & Girosi, F. (1990). Extensions of a theory of networks for approximation and learning: Dimensionality reduction and clustering. Tech. rep. AIM-1167, MIT-AILab.

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