Constructing Boosting Algorithms from SVMs: an Application to One-Class Classification

Gunnar Rätsch, Sebastian Mika, Bernhard Schölkopf, Klaus-Robert Müller

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 machine (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 base 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.**

Keywords—**Boosting, SVMs, one-class classification, unsupervised learning, novelty detection**

I. INTRODUCTION

Boosting methods have successfully been applied to classification problems [1], [2], [3], [4], [5], [6] and more recently also to regression estimation [7], [8], [9]. Their high accuracy, ease of implementation, and wide applicability has placed them in the standard toolbox of machine learning, next to neural networks [10], [11], [12] and kernel based learning methods like support vector machines (SVMs) [13], [14], [15], [16], [17], [18], [19], [20].

The present paper focuses on two points. On the algorithmic side we will propose a boosting-like *one-class classification* algorithm based on a technique called barrier optimization (standard books on optimization, e.g. [21]). In one-class classification one trains on unlabeled data, trying to assess whether a test point is likely to belong to the distribution underlying the training data (cf. [22], [23]). 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 [22], [23], [24], but not for boosting. In fact, and this is the second focus of this article, we will derive our boosting-like algorithms by using an equivalence between SVM and boosting on the level of the mathematical programs underlying both these algorithms. The general connection has already been described by e.g. Schapire et. al. [25], [26] and is nowadays considered a folklore statement in the boosting community. In the present work, we attempt to further elaborate on this insight by developing a general mechanism to convert SVM-like algorithms to boosting-like algorithms and vice versa.

A potential advantage of using a boosting-like one-class classifier 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, e.g., into a kernel function of SVM. 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 makes it possible to extract simple and interpretable rules from the decision boundary (cf. Section V).

The next section recapitulates some connections between boosting and SVMs and presents the general recipe to convert a SVM-like algorithm to a boosting-like algorithm. Using this relation, we first derive a linear programming formulation of the one-class problem in Section III and then develop a boosting-like technique in Section IV, building upon results of [8] that connect boosting to barrier optimization techniques. The experimental Section V shows the validity of our approach. Finally, we give some brief conclusions.

II. SVMS AND BOOSTING

In the next two subsections we will review some 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 I for a summary of the notation used in this paper.

A. Support Vector Machines

Consider a J-dimensional *feature space* $\mathcal{F}$ which is a subset of $\mathbb{R}^J$ (where $J \in \mathbb{N} \cup \{\infty\}$), and a mapping $\Phi : \mathcal{X} \to \mathcal{F}$. 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 $\mathcal{F}$:

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

Finding the hyperplane can be cast as a quadratic optimization problem [13]:

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

where for simplicity we have omitted the bias/offset. One selects the hyperplane from a set with small (VC) capacity [14], which in this case can be achieved by maximizing the margin. Here,

G. Rätsch (IEEE Member) is with the Australian National University, RSISE, Canberra, ACT 0200, Australia, EMail Gunnar.Raetsch@anu.edu.au, S. Mika, and K.-R. Müller are with Fraunhofer FIRST, Kekuléstr. 7, 12489 Berlin, Germany, EMail {Sebastian.Mika, Klaus-Robert.Mueller}@first.fraunhofer.de; K.-R. Müller is also with the University of Potsdam, Department of Computer Science, August-Bebelstr. 89, 14482 Potsdam, Germany; B. Schölkopf is with the Max-Planck-Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany, EMail Bernhard.Schoelkopf@tuebingen.mpg.de.
We thank for valuable discussions with Manfred Warmuth, Alex Smola, Bob Williamson, and Ayhan Demiriz. Thanks also to the anonymous referees for thorough reviews, valuable comments and suggestions that significantly improvement this work. This work was partially funded by DFG under contract JA 379/9-1, JA 379/7-1, 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.

the margin $\rho = \frac{2}{\|\mathbf{w}\|_2}$ is defined as the minimum ℓ_2-distance over all training points to the separating hyperplane.

More generally, the *margin* of an example $(\mathbf{x}_n, y_n)$ can be defined as the ℓ_q-distance of the pattern to a given separating hyperplane. A positive margin corresponds to a correct classification and the larger the margin, the larger the "confidence" [25], [27], [14] that the classifier correctly classifies the given input. 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 [28]) Let $\mathbf{z} \in \mathbb{R}^J$ be any point which is not on the plane $A = \{\tilde{\mathbf{z}} \mid \langle \tilde{\mathbf{z}}, \boldsymbol{\alpha} \rangle = 0\}$. Then for $p \in [1, \infty]$:

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

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

Therefore, maximizing the minimal ℓ_q-distance of the training patterns to the separating hyperplane is equivalent to solving the following optimization problem [28]:

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

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 [13] 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 the coefficient vector $\boldsymbol{\lambda}$ is usually sparse. However, this does not hold for the weight vector $\mathbf{w}$ in feature space.

B. Boosting

Now we consider boosting algorithms [29], [30], [31]. Here, we will in particular focus on the Arc-GV algorithm [32]. For AdaBoost [31] similar connections can be made (see the discussion in [8] and also [33]). 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

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

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. [31]). 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 [32]). 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. [31] and [32], respectively.

Arc-GV has been shown[1] asymptotically [32] to find the linear combination that solves a linear optimization problem (LP) which is commonly stated as:[2]

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

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) [36], [37], [38], [39], [35]. This number can be bounded by the number of patterns N (independently of the size of $\mathcal{H}$), but is usually much smaller.[3]

C. Connections between Boosting and SVMs

It is a common folklore statement and has already been pointed out e.g. in [25], [26] 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. In the following we will recall this connection, making it more explicit. We will show that SVMs and boosting 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 explicitly 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. [40]) 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 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 countable hypothesis set $\mathcal{H}$ implies a mapping Φ by

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

[1] This is also believed (cf. [34], [32], [8]) to be true for AdaBoost, if the problem is separable, i.e. $\rho > 0$, and one employs weighted minimization (for a discussion see [35]).

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

[3] The 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.

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.[4]

III. From One-Class SVMs to One-Class-LPs

A. One-Class-SVMs

Over the last years, SVMs have been generalized in various ways to deal with a number of different learning problems, e.g. also the problem of unsupervised learning, i.e., where the data are unlabeled [20], [41]. One could hold the view that unsupervised learning is essentially synonymous with density estimation, for, once a density has been estimated, all other statistical properties of the regularity underlying the data can readily be deduced. However, density estimation in high-dimensional spaces is known to be hard. Moreover, not all distributions necessarily possess a density. These considerations formed the motivation for the one-class SVM (1-SVM) approach of [22]. It deals with a problem which is easier than density estimation: the problem of (given data) estimating regions of high probability (e.g., the support or a certain quantile of a distribution). In a nutshell, the goal is to estimate a region which will contain a large fraction of the training data while keeping the value of some regularizer small.[5] In [22], this regularizer is a SV style large margin regularizer.

Geometrically, this amounts to finding 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:

$$\begin{array}{ll} \min\limits_{\mathbf{w}\in\mathcal{F},\boldsymbol{\xi}\in\mathbb{R}_+^N,\rho\in\mathbb{R}} & \frac{1}{2}\|\mathbf{w}\|_2^2 - \nu\rho + \frac{1}{N}\sum\limits_{n=1}^{N}\xi_n \\ \text{s.t.} & (\mathbf{w}\cdot\Phi(\mathbf{x}_n)) \geq \rho - \xi_n, n = 1,\ldots,N. \end{array} \tag{8}$$

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 regularization term $\|\mathbf{w}\|_2^2$ [14]. The parameter $\nu \in (0, 1]$ controls the trade-off and, as we will show below, the fraction of points in the estimated region.

A related approach (called support vector data description — SVDD) is proposed in [23], where one is not seeking 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 [22].

B. 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 induced by the hypothesis set, as the ℓ_1-norm used in (8) instead of the ℓ_2-norm implies sparsity in the solution vector $\mathbf{w}$ (see discussion in Section II-B). Thus, we obtain the following optimization problem:

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

where we have to fix $\|\mathbf{w}\|_1 = 1$ because the objective is — unlike (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 $\rho > 0$ [35].

It is furthermore worthwhile to observe that the solutions of (8) and (9) have the ν-property (shown for (8) in [22]):

Proposition 2: Assume the solution of (9) satisfies $\rho \geq 0$. Then the following statement holds:

$$\frac{1}{N}\sum_{n=1}^{N} \mathrm{I}(\mathrm{f}_\mathbf{w}(\mathrm{x}_\mathrm{n}) < \rho) \leq \nu \leq \frac{1}{\mathrm{N}}\sum_{\mathrm{n}=1}^{\mathrm{N}} \mathrm{I}(\mathrm{f}_\mathbf{w}(\mathrm{x}_\mathrm{n}) \leq \rho), \tag{10}$$

where $\mathrm{I}(\cdot)$ is the indicator function, taking the value 1 if its argument is true and 0 otherwise (cf. Appendix I).
Statement (10) is equivalent to the conjunction of the following two statements
(i) ν is at least as large as the fraction of outliers (i.e. training points $\mathbf{x}_n$ with $f_\mathbf{w}(\mathbf{x}_n) < \rho$).
(ii) ν is no larger than the fraction of training points which are not inside (i.e. outside, or on the boundary of) the estimated region (i.e. points $\mathbf{x}_n$ with $f_\mathbf{w}(\mathbf{x}_n) \leq \rho$).

The proof is analogous to the ones in [22], [42]:

Proof: Since ρ is a variable of the optimization problem, the solution in particular is optimal with respect to ρ and $\boldsymbol{\xi}$. We now fix $\mathbf{w}$ and vary ρ, observing how the value of the objective function changes. Start from some value of ρ larger than the optimal one. If we decrease ρ, the term $-\nu\rho$ increases proportionally to ν. On the other hand, the term $\frac{1}{N}\sum_{n=1}^{N}\xi_n$ in (9) will decrease proportionally to the fraction of points that have a nonzero ξ_n (the outliers), since all these ξ_n values can be decreased by the same amount as ρ while still satisfying the constraints. As long as the fraction of outliers is greater than ν, it pays off to decrease ρ, since in this case, the effect of the second term is larger than the first one. Therefore, also at the optimum, we know that the fraction of outliers is no more than ν.

[4] A similar line of reasoning is given in [25], [26]. Here, the ℓ_1-norm in Boosting is motivated by a bound on the generalization error, which relies on the fact that the examples in feature space have small ℓ_∞-norm. In SVMs, however, one assumes that there exists a small ℓ_2-ball around the data. One then exploits this fact for corresponding generalization error bounds.

[5] For a suitably chosen regularizer, this will then imply that the region also contains a certain fraction of the *test* data; see [22] for details.

Typical applications of such one-class approaches are in novelty detection and condition monitoring. In applications of the latter type, it is often the case that negative data (e.g., measurements of a power plant which has left the "normal" range) are very rare, expensive, or even impossible, in which case it is desirable to learn a description from the positive data only, which can later be used to assess whether a novel point is likely to have been generated by the same process as the training data.

On the other hand, if we increase ρ, starting from some small value, we will have to correspondingly increase all ξ_n which either are nonzero *or* which correspond to points that are precisely at the boundary (i.e., for which $(\mathbf{w} \cdot \Phi(\mathbf{x}_n)) = \rho$). By the same argument as above, we see that at the optimum, the fraction of points which are either outliers or on the boundary of the estimated region will be at least ν. ■

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 (e.g., even if we are given validation data, as long as these are unlabeled, it is unclear how to trade off the size of the estimated region with the fraction of validation points that lie inside it). In practice, the parameter ν can often be selected from the knowledge about the problem at hand (see also Section V-C).

A similar algorithm has been proposed independently by [24], 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.

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 $h_j(\mathbf{x}_n) \in [-1, 1]$ (as usual) and that the hypothesis space is complementation closed. Then we can enforce $w_j \geq 0$ $(j = 1, \ldots, J)$ without changing the problem and get the following linear optimization problem:

$$\begin{array}{ll} \min\limits_{\substack{\mathbf{w} \in \mathbb{R}_+^J, \boldsymbol{\xi} \in \mathbb{R}_+^N, \\ \rho \in \mathbb{R}}} & -\nu\rho + \frac{1}{N} \sum\limits_{n=1}^{N} \xi_n \\ \text{s.t.} & \sum_{j=1}^{J} w_j h_j(\mathbf{x}_n) \geq \rho - \xi_n, \\ & \qquad n = 1, \ldots, N \\ & \|\mathbf{w}\|_1 = 1, \end{array} \tag{11}$$

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

IV. 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 [43], it works very similar to AdaBoost. However, being pedantic about not confusing the terms, we will use "leveraging" first introduced in [44] instead of "boosting".

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 hypothesis coefficients that have already been selected in last iterations, as proposed by [45], [46], [9].

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 [32], [47], [48], [39]. It has been shown that this relates to coordinate descent methods [49], [50], [35], barrier optimization techniques [8], [35] and to the Bregman algorithm [51], [52], [53], [35].

In the next section we consider as an example two leveraging approaches for each of these two categories and adapt them to the one-class problem. However, it will turn out that both of these approaches have certain drawbacks. Finally, in Section IV-B we will develop our new method which tries to combine the advantages of the two approaches while avoiding their shortcomings.

A. ν-Arc and Column-Generation for AdaBoost

Note that the formulation in Eq. (11) is very similar to the problems underlying ν-Arc [42] and Column-Generation-AdaBoost (CG-AdaBoost) [9]. 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 one-class boosting-like approach is based on a reformulation of (11) as a min-max problem. Such problems can be solved by the ν-Arc algorithm which, in turn, employs Arc-GV. The second approach is based on column-generation, a technique borrowed from mathematical programming. We do not go into much detail in this section since the proposed methods primarily serve as a motivation for the barrier approach proposed in Section IV-B.

The first approach — one-class ν-Arc — is based on the observation that the Arc-GV algorithm used by ν-Arc maximizes the minimum margin. The maximum margin problem is a min-max problem. Following the ideas in [42], one reformulates (11) as a *min-max-problem*:

$$\begin{array}{ll} \max\limits_{\mathbf{w} \in \mathbb{R}^J, \boldsymbol{\xi} \in \mathbb{R}_+^N, \rho \in \mathbb{R}} & \tilde{\rho} \\ \text{with} & \sum_{j=1}^{J} w_j h_j(\mathbf{x}_m) + \xi_m \\ & \quad - \frac{1}{\nu N} \sum\limits_{n=1}^{N} \xi_n \geq \tilde{\rho}, \\ & \qquad m = 1, \ldots, N \\ & \|\mathbf{w}\|_1 = 1, \end{array} \tag{12}$$

which has the same solution as (11). The idea of ν-Arc as presented in [42] for supervised learning, is to use Arc-GV to solve a version of (12) that uses labels, motivated by the fact that it is able to solve the particular min-max problem (6). In order to do this, we proposed to replace all occurrences of the hypothesis output $\sum_{j=1}^{J} w_j h_j(\mathbf{x}_m)$ in Arc-GV by another quantity (a soft margin) [39], [35]. Applied to the current case, one would use the left hand side of the constraints in (12), which are continually given to the Arc-GV algorithm (for details, especially on ν-Arc, see [42] and [35]). Then 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 (12) for large hypothesis classes. However, it is not known whether such an algorithm always converges to the optimal solution of (12).

The Column-Generation approach for AdaBoost [9] starts with the dual problem of a linear program that is similar to (11) and (6) and uses a technique well-known in the optimization community — Column Generation (e.g. [54]). Let us just briefly recapitulate this algorithm and show how it could be applied to our modified problem. The dual of (11) is (cf. Appendix III-B for the derivation)

$$\begin{array}{lrcll} \max\limits_{\mathbf{d}\in\mathbb{R}^N,\epsilon\in\mathbb{R}} & \epsilon & & & \\ \text{with} & \sum_{n=1}^N d_n h_j(\mathbf{x}_n) & \leq & \epsilon, & j=1,\dots,J \\ & 0 \leq d_n & \leq & \frac{1}{N}, & n=1,\dots,N \\ & \sum_{n=1}^N d_n & = & \nu, & \end{array} \tag{13}$$

where one has one constraint per hypothesis, box constraints on $\mathbf{d}$ in the N dimensional simplex and one equality constraint. The algorithm iteratively selects a hypothesis[6] h_j (column j) and constructs subsets $\mathcal{H}^t = \{h_1,\dots,h_t\} \subseteq \mathcal{H}$, $t=1,2,\dots$, for which it solves the problem (13) 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 (13) it can easily be seen that the hypothesis of choice is the one that makes most progress, i.e. the one with largest $\sum_{n=1}^N d_n h_j(\mathbf{x}_n)$ — called the *edge* in supervised learning [32]. Clearly, if all hypotheses have been added, the problem is solved. Due to the smart selection rule and the sparseness of the solution, the convergence of this algorithm is often faster and at some point there is no hypothesis left which would change the current solution, i.e. for all $j=1,\dots,J$, we have that $\sum_{n=1}^N d_n h_j(\mathbf{x}_n) \leq \epsilon$. 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 [55] and [35].

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 guarantees convergence and has, empirically, fast performance [9], but it is impossible to implement without the availability of a (very) good LP-solver. Therefore, we will not go into further details of these algorithms although there might be situations in which they can be useful. Instead, we rather take them as motivation to develop in the following sections a different, new method based on the work in [8] using *barrier optimization techniques* combining both algorithms and most of their advantages while avoiding their disadvantages.

B. A One-Class Barrier Algorithm

In [8] we have found that boosting algorithms such as AdaBoost and Arc-GV can be explained by barrier optimization techniques (see also [35]). Here we are going to use the same techniques to solve the one-class problem and show the convergence of our algorithm.

Let us very briefly recall some basic statements about barrier optimization. The goal of barrier optimization is to find an optimal solution of the problem

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

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. [56], [57], [58]) is a particular useful choice [8] for our purposes,

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

where $\beta > 0$ denotes a penalty parameter. By finding a sequence of (unconstrained) minimizers to (14), 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. [56]). The conditions for convergence are even more relaxed as shown in the following:

Proposition 3 ([56]) Assume f and c_i are differentiable and convex functions. Let $\mathbf{x}^t$ be a $\delta_t \geq 0$ minimizer of

$$E_{\beta_t}(\mathbf{x}) = f(\mathbf{x}) + \beta_t \sum_{i=1}^m \exp\left[-\frac{c_i(\mathbf{x})}{\beta_t}\right], \tag{15}$$

i.e. $\|\frac{\partial}{\partial\mathbf{x}} E_{\beta_t}(\mathbf{x}^t)\| \leq \delta_t$. Then, for $\delta_t, \beta_t \stackrel{t\to\infty}{\longrightarrow} 0$ every limit point of $\{\mathbf{x}^t\}_{t\in\mathbb{N}}$ is a solution to (14).

Let us now apply the exp-barrier technique to problem (11). We get the following barrier functional:

$$\begin{aligned} E_\beta(\rho,\mathbf{w},\boldsymbol{\xi}) = -\nu\rho + \frac{1}{N}\sum_{n=1}^N \xi_n \\ + \beta\sum_{n=1}^N \left(\exp\left[\frac{-\xi_n}{\beta}\right] + \exp\left[\frac{\delta_n - \xi_n}{\beta}\right]\right), \end{aligned} \tag{16}$$

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. In order to reduce the number of variables to be optimized, we find the optimal slack variables ξ_n by minimizing (16) 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:

$$\xi_n(\beta,\mathbf{w},\rho) = \beta\log\left(1+\exp\left[\frac{\delta_n}{\beta}\right]\right) + \beta\log N, \tag{17}$$

which — as expected[7] — satisfies $\lim\limits_{\beta\to 0} \xi_n(\beta,\rho,\mathbf{w}) = \max(0,\delta_n)$. By plugging in (17) into (16) we obtain the following simplified barrier objective:

$$E_\beta(\rho,\mathbf{w}) = -\nu\rho + \frac{\beta}{N}\sum_{n=1}^N \log\left(1+\exp\left[\frac{\delta_n}{\beta}\right]\right) + \beta, \tag{18}$$

which does not contain the variables ξ_n anymore. We therefore have only $J+1$ variables left to optimize. Furthermore, this functional form is very similar to the loss used in logistic regression (cf. [11], [47], [53]; in our case it has just the additional offset ρ and the scaling factor β).

Usually, in a barrier algorithm one would optimize all $J+1$ parameters directly until the desired precision is reached

[6] 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.

[7] This becomes clear by (12) and Section IV-A.

(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 only needs a δ-minimizer for E_β. It will turn out that the base-learner helps to estimate the ℓ_∞-norm of the gradient.

B.1 Optimization on the Simplex

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 $|\mathbf{w}'|$ is the element-wise absolute value of $\mathbf{w}'$. Let us assume for now that $\mathbf{w}' \geq \mathbf{0}$ (then we can omit taking the absolute value). Let $f_{\mathbf{w}'} = \sum_{j=1}^J w'_j h_j$. Then we have $f_{\mathbf{w}'} = \|\mathbf{w}'\|_1 f_{\mathbf{w}}$ and get the following objective from (18)

$$E_\beta(\rho, \mathbf{w}') = -\nu\rho + \beta \sum_{n=1}^N \log\left(1 + \exp\left[\frac{\rho - \|\mathbf{w}'\|_1^{-1} f_{\mathbf{w}'}(\mathbf{x}_n)}{\beta}\right]\right) + \beta.$$

Lemma 4: If $w'_j > 0$, the derivatives $\frac{\partial E_\beta(\rho,\mathbf{w}')}{\partial w'_j}$ can be be computed as:

$$\frac{\partial E_\beta(\rho, \mathbf{w}')}{\partial w'_j} = \frac{1}{\|\mathbf{w}'\|_1} \sum_{n=1}^N \left(f_{\mathbf{w}}(\mathbf{x}_n) - h_j(\mathbf{x}_n)\right) d_n, \qquad (19)$$

where

$$d_n = \frac{\exp\left[\frac{\rho - f_{\mathbf{w}}(\mathbf{x}_n)}{\beta}\right]}{1 + \exp\left[\frac{\rho - f_{\mathbf{w}}(\mathbf{x}_n)}{\beta}\right]}. \qquad (20)$$

For the proof see Appendix II-A.

At $w'_j = 0$ the function is not differentiable and one needs to use sub-gradients. However, to make the notation not overly heavy, we try to avoid them in this presentation and only note, that it is possible to make the derivations exact.[8] Also note, if $w'_j < 0$, the gradient with respect to w'_j is as in (19), where the sign of $h_j(\mathbf{x}_n)$ is flipped.

For simplicity we will use $\mathbf{w}$ instead of $\mathbf{w}'$ for the next sections, while having in mind that we use the above mentioned projection into the simplex. We will always evaluate the gradient at a weight vector $\mathbf{w}$ that satisfies $\|\mathbf{w}\|_1 = 1$ and $\mathbf{w} \geq 0$. We can therefore apply Lemma 4 and omit the term $\|\mathbf{w}'\|_1^{-1}$ in front of the sum in (19) when computing the gradient.

We conclude this section with the following:

Lemma 5: Suppose $\mathbf{w} \in \mathbb{R}_+^J$ with $\|\mathbf{w}\|_1 = 1$ and $\rho \in \mathbb{R}$ satisfy $\frac{\partial E_\beta(\rho,\mathbf{w})}{\partial w_j} \geq 0$ for all $j = 1, \ldots, J$ and $\frac{\partial E_\beta(\rho,\mathbf{w})}{\partial \rho} = 0$. Then $(\mathbf{w}, \rho)$ is a global minimizer of E_β on the probability simplex.

The proof directly follows from Appendix III-C, showing that the duality gap is zero in this case. Since $\mathbf{w}$ and ρ are feasible, they must also be optimal.

[8] This particularly concerns (29) and (32).

B.2 Selecting the Hypothesis

The gradient of E_β with respect to each w_j can be computed as (cf. Lemma 4):

$$\frac{\partial}{\partial w_j} E_\beta(\rho, \mathbf{w}) = \sum_{n=1}^N d_n(f_{\mathbf{w}}(\mathbf{x}_n) - h_j(\mathbf{x}_n)), \qquad (21)$$

where the d_n's are as in (20). To reduce E_β iteratively (for some fixed β) one may choose the hypothesis h_t with the most negative gradient component, i.e.

$$h_t = \operatorname*{argmax}_{h \in \mathcal{H}} \sum_{n=1}^N d_n h(\mathbf{x}_n) = \operatorname*{argmax}_{h \in \mathcal{H}} \left(-\frac{\partial}{\partial w_h} E_\beta\right), \qquad (22)$$

where w_h denotes the weight for the hypothesis h. In (22) one finds the hypothesis that is most responsible for the size of the duality gap since $\max_{h\in\mathcal{H}}\left(-\frac{\partial}{\partial w_h} E_\beta\right)$ is the duality gap between the primal problem (18) and its dual (cf. Appendix III-C). It is therefore always non-negative.

The sum in (22) is often called the *edge* of the hypothesis h [32]. In supervised boosting techniques, maximizing the edge corresponds to minimizing the training error using the pattern weighting $\mathbf{d}$. In order to find the hypothesis in the unsupervised setting, the scalar-product between the weight vector and the hypothesis outputs needs to be maximized (cf. Section IV-D).

The selection rule (22) is related to the so-called Gauss-Southwell method [21], [59] on the full problem (see also [35]). Here one also selects a coordinate with largest gradient component. For our proof of convergence (shown later) we actually need to assume less on the base learner. It is allowed to return a hypothesis with some large, but not necessarily the largest edge. More formally, the base learner needs to return a hypothesis h_t that satisfies

$$\sum_{n=1}^N d_n(h_t(\mathbf{x}_n) - f_{\mathbf{w}}(\mathbf{x}_n)) \geq \tau\left(\max_{h\in\mathcal{H}} \sum_{n=1}^N d_n(h(\mathbf{x}_n) - f_{\mathbf{w}}(\mathbf{x}_n))\right), \qquad (23)$$

where τ is a continuous and strictly monotonically increasing function with $\tau(0) = 0$. For instance, this could be $\tau(x) = \delta x$ for some small $\delta > 0$. Then the base learner only needs to return a hypothesis with edge only slightly higher than the edge of the combined hypothesis. This is a hypothesis that has a positive left hand side in (23), as long as one has not reached optimality. This is indeed the least one can require from the base learner. If the base learner satisfies (23), we say it is τ-optimal.

There are in fact many other criteria that could be used to formulate conditions on the base learner. For a detailed analysis of conditions on the base learner to achieve different rates of convergence see [35] (see also [59], [53], [50]). Obviously, if $\tau(x) = x$, then one would expect the fastest convergence. However, we are not going to exploit this fact here and consider the general τ-optimality condition as defined above for arbitrary functions τ.

B.3 Finding the Weights

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

$$\mathbf{w}^t = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{S}^t(\mathbf{w}^{t-1})} E_\beta(\rho,\mathbf{w}). \quad (24)$$

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, i.e.

$$\mathcal{S}^t(\mathbf{w}^{t-1}) = \{(1-\lambda)\mathbf{w}^{t-1}+\lambda\mathbf{e}_t \mid \lambda\in[0,1]\}, \quad (25)$$

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. $\mathcal{S}^t(\mathbf{w}^{t-1}) = \mathcal{S}^t$ like in the CG approach) if the minimization in (24) can be implemented efficiently. This may improve the convergence speed in practice. However, in our analysis we only consider the simpler case.

If the first derivatives of E_β are bounded (which is the case for $\beta > 0$), one can lower bound the improvement of E_β when choosing the weights as in (24). We have the following result relating the improvement to the gradient with respect to the chosen weight:

Lemma 6 ([35]) Let $q : S \to \mathbb{R}$ be a strictly convex and twice continuously differentiable function on an open, convex set $S \subseteq \mathbb{R}$. Assume its derivative q' is Lipschitz continuous (i.e. $|q'(\lambda_1)-q'(\lambda_2)| \leq L|\lambda_1-\lambda_2|$ holds for all $\lambda_1,\lambda_2\in S$ and for some fixed $L>0$). Assume $\lambda^* = \operatorname{argmin}_{\lambda\in S} q(\lambda)\in S$ exists. For any $\lambda\in S$ we have that

$$q(\lambda)-q(\lambda^*) \geq \frac{|q'(\lambda)|^2}{4L}, \quad (26)$$

where $q'(\lambda) = \frac{\partial q(\lambda)}{\partial\lambda}$.
The proof is shown in Appendix II-B.

We can apply this lemma for the case where $S=(0,1)$, and $q(\lambda) = E_\beta(\rho,\mathbf{w}(\lambda))$ as in (25) and starting at $\lambda = 0$. Here λ is the variable to be optimized. If the gradient[9] is smaller than zero, i.e. if the base learner returns a hypothesis with large edge, then λ will be larger than zero and the left open boundary does not matter. The right boundary does not matter, if the minimizing λ is not 1. For this special case we can follow the proof of Lemma 6 and show that $q(\lambda)-q(\lambda^*) \geq \frac{1}{2}|q'(\lambda)|$. Hence, we can conclude that we can lower bound the progress in the loss function E_β in terms of the gradient of the selected hypothesis.

From the proof of Lemma 6, one can see that it is sufficient to approximate the solution of the minimization in (24) by $\lambda^*\approx\min(1,-\frac{q'(\lambda)}{2L})$ to achieve the same result. Then the computation becomes even simpler; however, one needs to know the Lipschitz constant L beforehand.[10]

We have now discussed all parts of our algorithm, which is summarized in Algorithm 1 as pseudo-code. The line marked with $*$ will become clear in the proof of our convergence theorem.

[9]Here, the gradient with respect to λ is meant. It is easily shown that it is the same as the gradient with respect to the normalized weight vector.

[10]For our particular case, it is possible to bound L in terms of β. However, since the one-dimensional minimization is implemented easily by standard line-search techniques we omit the details of this simplification.

Algorithm 1 The 1-Class-Leveraging algorithm

argument: Sample $X=\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$
Number of iterations T
Quantile Parameter $\nu\in(\frac{1}{N},1)$
Factor for decreasing barrier parameter $c\in(0,1)$
returns: Convex combination from $\mathcal{H}$ solving the one-class problem.
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**
      $\beta = c\beta$
    endif
    $\mathbf{w}^t = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{S}^t(\mathbf{w}^{t-1})} \ldots$
      $\ldots \sum_{n=1}^N \log\left(1+\exp\left[\frac{\rho^{t-1}-f_{\mathbf{w}}(\mathbf{x}_n)}{\beta}\right]\right)$
    $d_n^{t+1} = \dfrac{\exp\left[\frac{\rho^t-f_{\mathbf{w}^t}(\mathbf{x}_n)}{\beta}\right]}{1+\exp\left[\frac{\rho^t-f_{\mathbf{w}^t}(\mathbf{x}_n)}{\beta}\right]}$,
      with ρ^t such that $\sum_{n=1}^N d_n^{t+1}=\nu$
  endfor
  return $f=\sum_{t=1}^T w_t h_t$
end

C. Proof of Convergence

To show the convergence of our algorithm, we combine several results that are technically quite involved. First we consider the case of fixed β and show that the barrier loss E_β is minimized. Later we consider the question how one should decrease β and can eventually show the convergence of Algorithm 1 using Proposition 3.

Let us assume β is fixed. To show the convergence towards a minimum of E_β, we use the so-called *auxiliary function technique* as used in [60], [53], [35]. We define an auxiliary function $\mathcal{A}$ for a sequence of iterates $(\rho_t,\mathbf{w}^t)$, $t=1,2,\ldots$ as a continuous function satisfying

$$E_\beta(\rho_t,\mathbf{w}^t)-E_\beta(\rho_{t+1},\mathbf{w}^{t+1}) \geq \mathcal{A}(\rho_t,\mathbf{w}^t)\geq 0 \quad (27)$$
$$\mathcal{A}(\rho_t,\mathbf{w}^t)=0 \quad\Rightarrow\quad \rho_t \text{ and } \mathbf{w}^t \text{ are optimal.} \quad (28)$$

The following lemma shows that if one can lower bound the progress in decreasing E_β by the auxiliary function, one eventually must reach a point close to the solution in a finite number of steps. To show this we adapt a result of [53], [60]:

Lemma 7: Let $\mathcal{A}$ be an auxiliary function for the sequence $(\rho_1,\mathbf{w}^1),(\rho_2,\mathbf{w}^2),\ldots$. Assume the $\mathbf{w}^t$'s lie in the J-dimensional probability simplex and $\rho_t\in[-1,1]$. Then any limit point of the above sequence is a minimizer of E_β on the probability simplex.

Proof: By (27), $E_\beta(\rho_t, \mathbf{w}^t)$ is a non-increasing sequence bounded from below by $-\nu$. Therefore, the sequence of differences $E_\beta(\rho_t, \mathbf{w}^t) - E_\beta(\rho_{t+1}, \mathbf{w}^{t+1}) \geq 0$ converges to zero. By (27), $\mathcal{A}(\rho_t, \mathbf{w}^t)$ must also converge to zero. Since the iterates $(\rho_t, \mathbf{w}^t)$ lie in a compact space, the sequence must have at least one limit point $(\tilde{\rho}, \tilde{\mathbf{w}})$. By continuity of $\mathcal{A}$, we have $\mathcal{A}(\tilde{\rho}, \tilde{\mathbf{w}}) = 0$. Therefore, the limit point $(\tilde{\rho}, \tilde{\mathbf{w}})$ is optimal by (28). Since this holds true for any limit point, we have completed the proof of the lemma. ■

Next, we will show that one can find an auxiliary function for the sequence generated by our algorithm. By an application of Lemma 7 we will conclude that our algorithm will generate optimal solutions. We have:

Proposition 8: Assume $\beta > 0$ is fixed in Algorithm 1. Suppose the base learner finds in each iteration a hypothesis that satisfies the τ-optimality criterion as given in (23) for some admissible function τ. Then any limit point of the sequence $(\rho^t, \mathbf{w}^t)$ is a global minimizer of E_β on the simplex.

Proof: We have already shown that the progress can be bounded in terms of the negative gradient components. We therefore propose the following function that will be used to define an auxiliary function:

$$\mathcal{A}^\nabla(\rho, \mathbf{w}) = \frac{1}{\sqrt{J}} \sqrt{\sum_{j=1}^{J} \left[-\frac{\partial E_\beta(\rho, \mathbf{w})}{\partial w_j} \right]_+^2} \tag{29}$$

where $[\cdot]_+$ is the continuous function defined by $[x]_+ = \max(0, x)$. Let $q(\lambda) = E_\beta(\rho_t, (1-\lambda)\mathbf{w}^t + \lambda \mathbf{e}_{t+1})$ (cf. (25)) and L be the Lipschitz constant of $q(\cdot)$. Such constant exists, if there exists a Lipschitz-constant for $\nabla E_\beta(\rho_t, \cdot)$, but since $\nabla^2 E_\beta(\rho_t, \cdot)$ is bounded, there must exist such a constant. Hence, there exists some $L < \infty$. By Lemma 6 optimize over $\mathbf{w}^t$ (cf. (25)) and the τ-optimality of h_t, it holds that

$$\begin{aligned} & E_\beta(\rho_t, \mathbf{w}^t) - E_\beta(\rho_{t+1}, \mathbf{w}^{t+1}) \\ \geq\ & E_\beta(\rho_t, \mathbf{w}^t) - E_\beta(\rho_t, \mathbf{w}^{t+1}) \\ \geq\ & \frac{\left(\sum_{n=1}^N d_n^t (h_{t+1}(\mathbf{x}_n) - f_{\mathbf{w}^t}(\mathbf{x}_n))\right)^2}{4L}, \end{aligned} \tag{30}$$

if $\lambda < 1$ in (25). If $\lambda = 1$, then one may bound $E_\beta(\rho_t, \mathbf{w}^t) - E_\beta(\rho_{t+1}, \mathbf{w}^{t+1}) \geq \frac{1}{2} \sum_{n=1}^{N} d_n(h_t(\mathbf{x}_n) - f_{\mathbf{w}}(\mathbf{x}_n))$ and one could follow our derivations for this case. Here, we assume the more difficult case $\lambda < 1$.

If the base learner returns a hypothesis h_t satisfying (23), then it also satisfies

$$\sum_{n=1}^{N} d_n(h_t(\mathbf{x}_n) - f_{\mathbf{w}}(\mathbf{x}_n)) \geq \tau\left(\mathcal{A}^\nabla(\rho, \mathbf{w})\right), \tag{31}$$

since τ is strictly increasing and $\max_{h \in \mathcal{H}} \sum_{n=1}^N d_n(h(\mathbf{x}_n) - f_{\mathbf{w}}(\mathbf{x}_n)) \geq \mathcal{A}^\nabla(\rho, \mathbf{w})$.

Combining (30) and (31) yields

$$\begin{aligned} E_\beta(\rho_t, \mathbf{w}^t) - E_\beta(\rho_{t+1}, \mathbf{w}^{t+1}) \ &\geq\ \frac{1}{4L}\tau\left(\mathcal{A}^\nabla(\rho, \mathbf{w})\right)^2 \\ &=:\ \mathcal{A}(\rho, \mathbf{w}). \end{aligned}$$

Hence, we have constructed a function $\mathcal{A}$ that satisfies the first condition for auxiliary functions (cf. (27); the positivity is obviously satisfied).

In Algorithm 1, ρ_t is determined such that $\sum_{n=1}^N d_n^{t+1} = \nu$. It can easily be verified that such $\rho \in [-1, 1]$ exists, if $\nu \in (\frac{1}{N}, 1)$ and that for this case holds $\frac{\partial E_\beta(\rho, \mathbf{w})}{\partial \rho} = \sum_{n=1}^N d_n^{t+1} - \nu = 0$. Hence also the second condition (28) is satisfied, since $\frac{\partial E_\beta(\rho_t, \mathbf{w}^t)}{\partial \rho} = 0$ at the end of each iteration and $\mathcal{A}^\nabla(\rho, \mathbf{w}) = \mathcal{A}(\rho, \mathbf{w}) = 0$, if and only if $\frac{\partial E_\beta(\rho, \mathbf{w})}{\partial w_j} \geq 0$ (cf. Lemma 5).

Hence, $\mathcal{A}$ is an auxiliary function for the iterates generated by our algorithm. We can therefore apply Lemma 7 and conclude that any limit point of this sequence is a global minimizer of E_β. ■

Note that we have only used some basic properties of the functional form of E_β and that the proof holds also for other functions. A similar result has been obtained in [50], but relying on stronger assumptions on the base learner. Related results have been obtained in [35] for the optimization on a box-constraint sets of $\mathbf{w}$ (see also [59]).

The following theorem combines the previous results and shows that our algorithm converges towards a solution of (11), i.e. the original optimization problem the barrier approach is supposed to solve (cf. (16) and (18)):

Theorem 9: Suppose $\mathcal{H}$ is a complementation closed, bounded and finite hypothesis set and the base learner $\mathcal{L}_\mathcal{H}$, when called by Algorithm 1, returns a hypothesis that fulfills the τ-optimality (23). Then any limit point of the output of the algorithm is a solution of (11).

Proof: The idea of the proof is as follows: For fixed β the algorithm converges to a minimum of E_β by Proposition 8. Hence, the smallest gradient (duality gap) will converge to zero. If it is small enough, one reduces β by a factor $c \in (0, 1)$ (cf. step $*$ in Algorithm 1). Since this holds for any $\beta > 0$, we can conclude that $\beta \to 0$. We show that β is reduced only, if the gradient of the full problem (i.e. with respect to all $J+1$ variables) is small. Then, we will show that the gradients do not change drastically, if one reduces β. Hence also the gradients of the full problem must converge to zero. Then we can apply Proposition 3 to conclude that any limit point is an optimal solution of (11).

Let us first bound the gradient of the full problem in terms of the edge returned by the τ-optimal base learner. By (23), we have

$$\begin{aligned} & \tau^{-1}\left(\sum_{n=1}^{N} d_n(h_t(\mathbf{x}_n) - f_{\mathbf{w}}(\mathbf{x}_n))\right) \\ \geq\ & \max_{h \in \mathcal{H}} \sum_{n=1}^{N} d_n(h(\mathbf{x}_n) - f_{\mathbf{w}}(\mathbf{x}_n)) \\ \geq\ & \frac{1}{J} \sum_{j \in \mathcal{J}} \sum_{n=1}^{N} d_n(h_j(\mathbf{x}_n) - f_{\mathbf{w}}(\mathbf{x}_n)) \geq 0, \end{aligned} \tag{32}$$

where the sum is only taken over the hypotheses with indices $\mathcal{J}$ with negative gradient components, i.e. where the second sum in the last line is greater than zero (leading to a duality gap, cf. Appendix III-C). Hence, the full gradient is bounded by $J\tau^{-1}\left(\sum_{n=1}^N d_n(h_t(\mathbf{x}_n) - f_{\mathbf{w}}(\mathbf{x}_n))\right)$. By condition $*$ in Algorithm 1, the barrier parameter β is reduced if the gradient of the full problem is smaller than $J\tau^{-1}(\beta)$ (the function τ^{-1} is strictly monotonically increasing and continuous).

By Proposition 8 we can conclude that the gradient, let us call it $\nabla_- E_\beta$, is smaller than β after a finite number of iterations (for every $\beta > 0$). Hence, $\beta \to 0$. Furthermore, the gradients just before decreasing β are also converging to 0. Thus a limit point of this subsequence solves (11).

It is left to show that the gradient does change only slightly, when decreasing β by the factor c. This is necessary to show that any limit point solves (11). For this, we only need to consider the change in the pattern weighting $\mathbf{d}$ (cf. (19)), which is computed by

$$d_n^{t+1} = \left(1 + \exp\left[\frac{f_{\mathbf{w}^t}(\mathbf{x}_n) - \rho^t}{\beta}\right]\right)^{-1}.$$

Consider the following Taylor expansion of the weight, when changing β by Δ:

$$\begin{aligned}\frac{1}{1+\exp(a/(\beta+\Delta))} &= \frac{1}{1+\exp(a/\beta)} \\ &+ \frac{a\exp(a/\beta)}{(1+\exp(a/\beta))^2\beta^2}\Delta + \mathcal{O}(\Delta^2)\end{aligned}. \quad (33)$$

Thus, for small enough β, when reducing β by a factor of c we have that $d_n^{(\beta)}$ is changing only little

$$\begin{aligned} d_n^{(c\beta)} &\approx d_n^{(\beta)} + \frac{c_1 a_n}{\beta}\frac{\exp(a_n/\beta)}{(1+\exp(a_n/\beta))^2} \\ &\approx d_n^{(\beta)} + \frac{c_1 a_n}{\beta}\cdot\begin{cases}\frac{1}{1+\exp(a_n/\beta)} & a_n > 0\\ 1/2 & a_n = 0\\ \exp(a_n/\beta) & a_n < 0\end{cases}\end{aligned}$$

where $c_1 = 1 - c$ and $a_n = f_{\mathbf{w}}(\mathbf{x}_n) - \rho^t$. Since $\exp(-|a_n|/\beta)$ decreases much faster than β^{-1}, we have for either $a_n = 0$ or for $|a_n| > 0$ with β small enough: $d_n^{(c\beta)} \approx d_n^{(\beta)}$. Thus, asymptotically, if the gradient is small for β, it is also for $c\beta$. We can conclude: $\lim_{t\to\infty}\|\nabla_- E_\beta(\rho^t, \mathbf{w}^t)\|_\infty = 0$.

Hence, we can apply Proposition 3 and conclude that any limit point of the sequence generated by our algorithm is an optimal solution of (11). ■

D. *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 depends of course on the problem to be solved. 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 themselves 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.[11] 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. (22)). 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

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

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

D.1 Sparse Combinations ($p = 1$)

For this case the minimization of $E(\boldsymbol{\alpha})$ has a very simple solution. It has been shown [55] 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 the maximum $g_q = \max_{q'} g_{q'}$ is assumed to be 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 smaller/simpler hypothesis set $\mathcal{H} = \{k_q \mid q = 1, \ldots, Q\}$ and obtains the same solution.

D.2 Least Squares ($p = 2$)

Under the assumption that $\|h\|_2 = \sqrt{\sum_n h(\mathbf{x}_n)^2}$ is constant for all $h \in \mathcal{H}$, it has been pointed out [8], [9] that (22) 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

$$\begin{aligned}\boldsymbol{\alpha} &= \operatorname*{argmin}_{\boldsymbol{\alpha}} E(\boldsymbol{\alpha}) \\ &= \operatorname*{argmin}_{\boldsymbol{\alpha}} \left[\sum_{n=1}^N \left((d_n - h_{\boldsymbol{\alpha}}(\mathbf{x}_n))^2 - h_{\boldsymbol{\alpha}}(\mathbf{x}_n)^2\right) + C\|\boldsymbol{\alpha}\|_2^2\right],\end{aligned}$$

where we use $\|\boldsymbol{\alpha}\|_2^2$ as a regularizer that effectively bounds the ℓ_2-norm of $\boldsymbol{\alpha}$ (cf. [62]). 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.

[11] In [55] the concept of *active kernels* has been proposed (also known as *moving centers* in [61]), 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. [55]). Here, we restrict us to the finite hypothesis case.

D.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:

$$\mathcal{H}^s = \{h_{c,i}(\mathbf{x}) = \tanh((x_n - c)s) \mid c \in \{\text{set of possible cuts}\}, i = 1, \ldots, \dim(\mathrm{X})\} \quad (34)$$

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.

V. Experiments

In the following experiments we will illustrate that our algorithms do what they should do — detecting novelty, anomalies 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.

A. 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 (250 patterns). This leads to a large density of patterns in the center, and to a lower density in the periphery. In this toy model, the latter are considered as outliers.

We used decision stumps of the form described in Section IV-D with $s = 4$ (there are 1000 different stumps). Figure 1 (4th column) shows the result for the barrier algorithm (cf. Algorithm 1) for $T = 100$ and $c = 0.7$. The 5th column in Figure 1 shows the result when directly solving the linear programming problem (11) on all stumps using a simplex method (which becomes intractable when using much more examples or using higher dimensional data). It illustrates how the parameter ν (from top ($\nu = 0.1$) to 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. Note that the optimal solution of (11) uses only these four stumps for the combined hypothesis. The barrier algorithm first selects more than four stumps, but then swiftly reduces the weight of all stumps except the four to zero (for instance, after 100 iterations, the weight of two other stumps have been about 10^{-15} and 10^{-10}).

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}. The plots show 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.[12]

B. Experiment on USPS database

In a second 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. [15]). Figure 2 shows a plot of the outputs on training 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 [22] for the 1-SVM, with comparable results.

C. Model Selection

A serious problem in the one-class approach – as for every unsupervised learning technique – 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 heuristic that may help to find an appropriate ν.

The idea is as follows: For any ν, the one-class-classifier will find some boundary irrespectively of the quality of separation between the set of accepted and rejected patterns. Intuitively, a good separation is achieved, if it is very clear, i.e. the distance between the two sets is large. We propose to use exactly this intuition as a criterion for model-selection. For given ν one measures the average output of the classifier for both sets and selects the ν that separates 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.

[12]By the dual of the barrier optimization problem (16) given in Appendix III-A, 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. [8] for some more details). Note that, differently from the LP-approach (which is favoring sparse patterns weights $\mathbf{d}$), the barrier approach tries to avoid this sparseness by using an exponential form (cf. (35)).

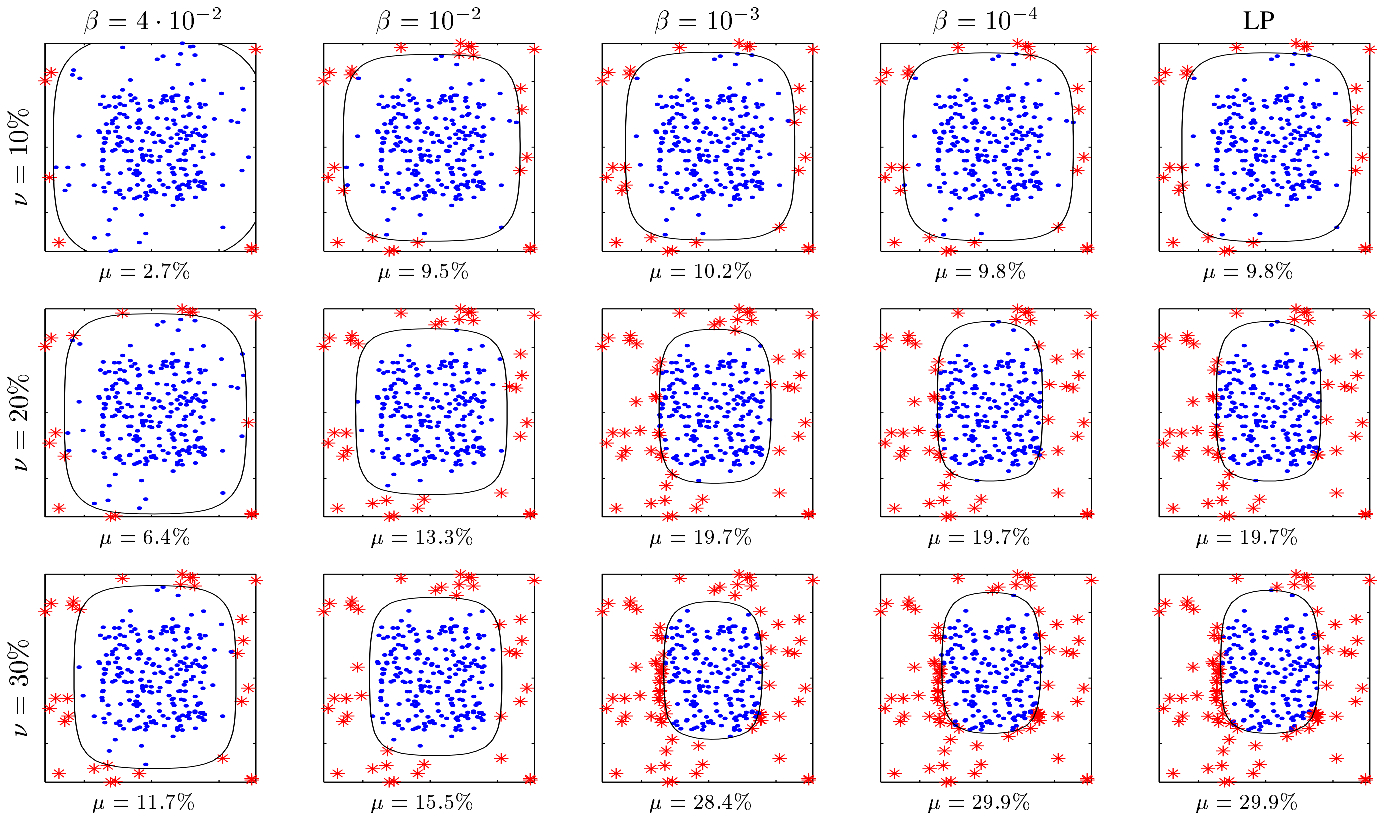


Fig. 1. The One-Class approach using tanh-decision stumps for finding outliers of this toy problem for different values of ν (10% (top line), 20% (middle line) and 30% (bottom line)) for different values of β in our one-class barrier algorithm (first 4 columns) and for the one-class LP (right most column). 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.

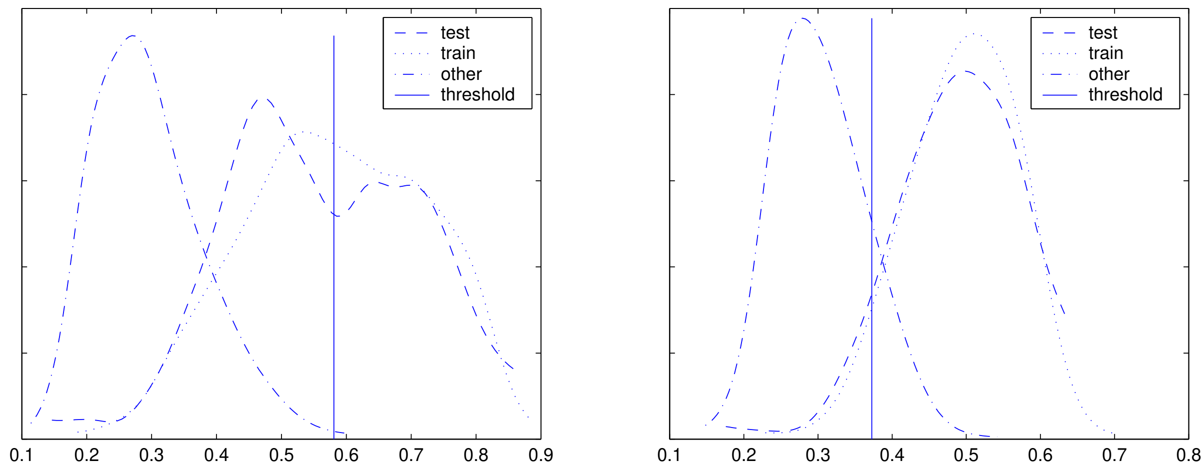


Fig. 2. Experiments on the US postal service OCR data set. Recognizer for digit 0; output histogram for the exemplars of 0 in the training/test set, and on test exemplars of other digits. The x-axis gives the output values, i.e. the argument of the sign in the decision function. For $\nu = 50\%$ *(left)*, we get 49.9% outliers (consistent with Prop. 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.

For each of these data sets, we compute the solution of the one-class-LP (9) for different values of ν.[13] 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 sets) and chose the $\nu \in (0,1)$ with largest separation distance. In our example the maximum

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

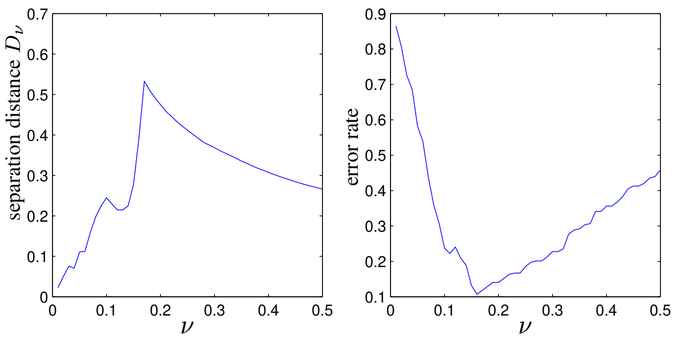


Fig. 3. Digit "1" against the others: On a data set with 1005 "1"-digits and 120 randomly sampled other digits, the one-class-LP classifier was applied. The separation distance D_ν (left) and the error rate of misclassification on a labeled test set (right) for different values of ν is shown. The maximum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

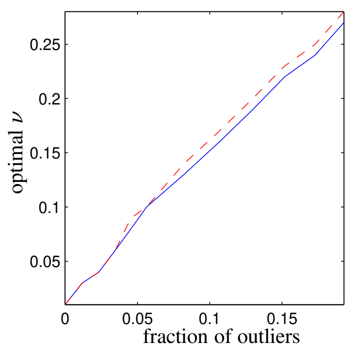


Fig. 4. Data set with 1005 "1"-digits and different fractions (x-axis) of randomly sampled other digits: The dashed graph shows the ν selected by our maximum-separation criterion. The solid graph shows the ν that minimizes the test-error.

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 true data and outliers.

VI. Conclusion

We have studied the correspondence between SVMs and boosting-like methods. 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 base hypotheses. The problem becomes tractable since an effective ℓ_1-norm regularizer is used. This induces sparsity, hence we never really work in the full space, but always in a small subspace. On the other hand, 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 explicitly work in the feature space. Therefore, SVMs get away without having to use ℓ_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. In fact, it gives rise to a simple recipe to transfer algorithms: *the SV-kernel is replaced by an appropriately constructed hypothesis space for leveraging where the optimization of an analogous mathematical program is done using ℓ_1 instead of ℓ_2 norm.* Vice versa, given a hypothesis set, it could be translated into a SV-kernel by Eq.(7). Here, we have employed the recipe 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 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. [63], [64]).

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.

Appendix

I. 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 $\mathbf{x}_n$
$f_{\mathbf{w}}$	a 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 $\mathcal{F}$
$\mathrm{I}(\cdot)$	indicator function ($\mathrm{I}(\mathrm{true}) = 1$ and $\mathrm{I}(\mathrm{false}) = 0$)

II. PROOFS

A. Proof of Lemma 4

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

where d_n is given as in (20) and we have assumed $w_j > 0$.

B. Proof of Lemma 6

Let $\beta^* = \lambda^* - \lambda$. We have $q'(\lambda^*) = 0$ since λ^* is optimal and $q'(\lambda) q'(\lambda + \beta) \geq 0$ for all β such that $\beta\beta^* \geq 0$ and $|\beta| \leq |\beta^*|$. Thus

$$
\begin{aligned}
|q(\lambda) - q(\lambda + \beta)| &= \left| \int_{\lambda}^{\lambda+\beta} q'(\tilde{\lambda}) d\tilde{\lambda} \right| \\
&= \operatorname{sign}(\beta) \int_{\lambda}^{\lambda+\beta} |q'(\tilde{\lambda})| d\tilde{\lambda}.
\end{aligned}
$$

Since q is strictly convex, q' is strictly monotone. Hence

$$
\begin{aligned}
&\operatorname{sign}(\beta) \int_{\lambda}^{\lambda+\beta} |q'(\tilde{\lambda})| d\tilde{\lambda} \\
\geq\ & \operatorname{sign}(\beta) \int_{\lambda}^{\lambda+\beta} |q'(\lambda + \beta)| d\tilde{\lambda} \\
=\ & \beta \operatorname{sign}(\beta) |q'(\lambda + \beta)| \\
=\ & |\beta|\, |q'(\lambda + \beta)|.
\end{aligned}
$$

Moreover, we have by optimality and assumption

$$
|q'(\lambda) - q'(\lambda^*)| = |q'(\lambda)| \leq L|\beta^*|.
$$

Thus, $|\beta^*| \geq \frac{|q'(\lambda)|}{L}$. Let $\hat{\beta}$ such that $\hat{\beta}\beta^* \geq 0$ and $|\hat{\beta}| = \frac{|q'(\lambda)|}{2L}$. From the Lipschitz differentiability holds

$$
|q'(\lambda) - q'(\lambda + \hat{\beta})| \leq L|\hat{\beta}| = \frac{|q'(\lambda)|}{2}.
$$

Since $q'(\lambda)$ and $q'(\lambda + \hat{\beta})$ have the same sign and $|q'(\lambda)| \geq q'(\lambda + \hat{\beta})$, follows

$$
|q'(\lambda + \hat{\beta})| \geq \frac{1}{2} |q'(\lambda)|.
$$

We now combine the results:

$$
\begin{aligned}
q(\lambda) - q(\lambda^*) &\geq q(\lambda) - q(\lambda + \hat{\beta}) \\
&= |q(\lambda) - q(\lambda + \hat{\beta})| \\
&\geq |\hat{\beta}|\, |q'(\lambda + \hat{\beta})| \\
&\geq \frac{1}{2} |\hat{\beta}|\, |q'(\lambda)| = \frac{|q'(\lambda)|^2}{4L}.
\end{aligned}
$$

■

III. PRIMAL/DUAL PROBLEMS

A. Duality for the Barrier Problem

Let us start with the following problem, for which we show that it is the dual optimization problem of (16) and (13):

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

where we use $A_{nj} = h_j(\mathbf{x}_n)$ for simplicity. By A_n we denote the n-th row of A. The Lagrangian for this problem is

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

where we introduced to sets of Lagrange multipliers, say $\mathbf{w}$ and $\boldsymbol{\xi}$, and we require $\mathbf{w} \geq \mathbf{0}$. Note that since $\mathbf{d}$ and $\mathbf{q}$ are always positive, one may add additional constraints $\mathbf{d}, \mathbf{q} \geq \mathbf{0}$ without changing the solution. Setting the derivatives with respect to ϵ, $\mathbf{d}$ and $\mathbf{q}$ variables to zero, yields $\sum_{j=1}^{J} w_j = 1$ and two non-linear constraint-sets

$$
d_n = \exp\left(\frac{\rho - \xi_n - A_n \mathbf{w}}{\beta} \right) \text{ and } q_n = \exp\left(\frac{-\xi_n}{\beta} \right). \quad (35)
$$

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

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

and we get the dual program as stated in (16):

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

B. Duality for the Linear Program

To derive the dual of (9), one could follow the same derivations as for the last optimization problem. For brevity we omit a detailed derivation and just show that the (9) is obtained by letting β go to zero in (18). For this consider the terms in the sum of (18):

$$
g_\beta(f_{\mathbf{w}}(\mathbf{x}_n)) = \beta \log \left(1 + \exp\left[\frac{\rho - f_{\mathbf{w}}(\mathbf{x}_n)}{\beta} \right] \right).
$$

It can easily be verified that g_β converges to $\max(0, \rho - f_{\mathbf{w}}(\mathbf{x}_n))$ for $\beta \to 0$. Hence, for $\beta \to 0$, the problems (9) and (18) (s.t. $\mathbf{w} \geq 0$ and $\sum_{j=1}^J w_j = 1$) are equivalent. We may conclude that the dual problems are also equivalent for $\beta \to 0$ and arrive at the dual problem of (9):

$$\begin{aligned} \min \quad & \epsilon \\ \text{s.t.} \quad & \langle \mathbf{d}, \mathbf{1} \rangle = \nu,\ 0 \leq d_n \leq \frac{1}{N} \\ & A^\top \mathbf{d} \leq \epsilon \mathbf{1}. \end{aligned}$$

Here, we removed the variables $\mathbf{q} \geq \mathbf{0}$ and rewrote the equality constraint $d_n + q_n = \frac{1}{N}$ as an inequality constraint.

C. Duality Gap

The duality gap is the difference between primal and dual objective for some given primal and dual variables. If all variables are feasible and the duality gap is zero, one has reached an optimal primal and dual solution. We compute the duality gap for a given $\mathbf{w}$, where $\mathbf{d}$ and $\mathbf{q}$ are given as in (35). Then, it can easily be seen that the variable ϵ is feasible and optimal for $\epsilon = \max_j \sum_n d_n A_{nj}$. Then the duality gap G is:

$$G(\mathbf{w}) = \underbrace{\beta \sum_{n=1}^N [d_n(\log d_n - 1) + q_n(\log q_n - 1)] + \epsilon}_{\text{dual objective}} \quad (36)$$

$$\underbrace{-\nu\rho + \frac{1}{N}\langle \boldsymbol{\xi}, \mathbf{1} \rangle + \beta \sum_{n=1}^N [d_n + q_n]}_{\text{(negative) primal objective}}$$

$$= \beta \sum_{i=1}^N d_n(\rho - \xi_n - A_n\mathbf{w}) \quad (37)$$

$$- \sum_{n=1}^N q_n \xi_n + \epsilon + \nu\rho - \frac{1}{N}\langle \boldsymbol{\xi}, \mathbf{1} \rangle$$

$$= \max_{j=1,\ldots,J} \sum_n d_n(A_{nj} - A_n\mathbf{w}) \geq 0. \quad (38)$$

The last line is equivalent to $\max_{j=1,\ldots,J} \left(-\frac{\partial E_\beta(\rho, \mathbf{w})}{\partial w_j} \right) \geq 0$.

REFERENCES

[1] H. Drucker, R. Schapire, and P.Y. Simard, "Boosting performance in neural networks," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, pp. 705–719, 1993.

[2] Y.A. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J.S. Denker, H. Drucker, I. Guyon, U. A. Müller, E. Säckinger, P.Y. Simard, and V.N. Vapnik, "Comparison of learning algorithms for handwritten digit recognition," in *Proc. ICANN'95*, F. Fogelman-Soulié and P. Gallinari, Eds., Nanterre, France, 1995, vol. II, pp. 53–60.

[3] R. Maclin and D. Opitz, "An empirical evaluation of bagging and boosting," in *Proceedings of the Fourteenth National Conference on AI*, 1997, pp. 546–551.

[4] H. Schwenk and Y. Bengio, "AdaBoosting neural networks," in *Proc. ICANN'97*, W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, Eds. 1997, vol. 1327 of *LNCS*, pp. 967–972, Springer.

[5] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithm: Bagging, boosting and variants," *Machine Learning*, vol. 36, pp. 105–142, 1999.

[6] T.G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, 1999.

[7] N. Duffy and D.P. Helmbold, "Leveraging for regression," in *Proc. COLT*, San Francisco, 2000, pp. 208–219, Morgan Kaufmann.

[8] G. Rätsch, M. Warmuth, S. Mika, T. Onoda, S. Lemm, and K.-R. Müller, "Barrier boosting," in *Proc. COLT*, San Francisco, 2000, pp. 170–179, Morgan Kaufmann.

[9] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *J. of Mach. Learning Research*, 2001, To appear in special issue on Support Vector Machines and Kernel Methods.

[10] S. Haykin, *Neural Networks : A Comprehensive Foundation*, Macmillan, New York, 1994.

[11] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[12] G. Orr and K.-R. Müller, Eds., *Neural Networks: Tricks of the Trade*, vol. 1524, Springer LNCS, 1998.

[13] B.E. Boser, I.M. Guyon, and V.N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, D. Haussler, Ed., 1992, pp. 144–152.

[14] V.N. Vapnik, *The nature of statistical learning theory*, Springer Verlag, New York, 1995.

[15] B. Schölkopf, *Support vector learning*, Oldenbourg Verlag, Munich, 1997.

[16] V.N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[17] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, Cambridge, UK, 2000.

[18] B. Schölkopf and A. J. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.

[19] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition," *Knowledge Discovery and Data Mining*, vol. 2, no. 2, pp. 121–167, 1998.

[20] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 181–201, 2001.

[21] D.G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley Publishing Co., Reading, second edition, May 1984,

[22] B. Schölkopf, J. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[23] D. Tax and R. Duin, "Data domain description by support vectors," in *Proc. ESANN*, M. Verleysen, Ed., Brussels, 1999, pp. 251–256, D. Facto Press.

[24] C. Campbell and K.P. Bennett, "A linear programming approach to novelty detection," in *Adv. in Neural Information Processing Systems*, T.K. Leen, T.G. Dietterich, and V. Tresp, Eds. 2001, vol. 13, pp. 395–401, MIT Press.

[25] R.E. Schapire, Y. Freund, P.L. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The Annals of Statistics*, vol. 26, no. 5, pp. 1651–1686, October 1998.

[26] Y. Freund and R.E. Schapire, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, September 1999, Appeared in Japanese, translation by Naoki Abe.

[27] R.E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," in *Proc. COLT'98*, 1998, pp. 80–91.

[28] O.L. Mangasarian, "Arbitrary-norm separating plane," *Operation Research Letters*, vol. 24, no. 1, pp. 15–23, 1999.

[29] L.G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, Nov. 1984.

[30] R.E. Schapire, *The Design and Analysis of Efficient Learning Algorithms*, Ph.D. thesis, MIT Press, 1992.

[31] Y. Freund and R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *EuroCOLT: European Conference on Computational Learning Theory*. 1994, LNCS.

[32] L. Breiman, "Prediction games and arcing algorithms," *Neural Computation*, vol. 11, no. 7, pp. 1493–1518, 1999, Also Technical Report 504, Statistics Department, University of California Berkeley.

[33] G. Rätsch and M.K. Warmuth, "Marginal boosting," NeuroCOLT2 Technical Report 97, Royal Holloway College, London, July 2001.

[34] Y. Freund and R. Schapire, "Game theory, on-line prediction and boosting," in *Proc. COLT*, New York, NY, 1996, pp. 325–332, ACM Press.

[35] G. Rätsch, *Robust Boosting via Convex Optimization*, Ph.D. thesis, University of Potsdam, 2001, http://mlg.anu.edu.au/~raetsch/thesis.ps.gz.

[36] K.P. Bennett and O.L. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets," *Optimization Methods and Software*, vol. 1, pp. 23–34, 1992.

[37] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. of Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1999.

[38] P. Bradley, O. Mangasarian, and J. Rosen, "Parsimonious least norm approximation," *Computational Optimization and Applications*, vol. 11, no. 1, pp. 5–21, 1998.

[39] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for AdaBoost," *Ma-*

chine Learning, vol. 42, no. 3, pp. 287–320, Mar. 2001, also NeuroCOLT Technical Report NC-TR-1998-021.

[40] O.L. Mangasarian, “Mathematical programming in data mining,” *Data Mining and Knowledge Discovery*, vol. 42, no. 1, pp. 183–201, 1997.

[41] B. Schölkopf, A.J. Smola, and K.-R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[42] G. Rätsch, B. Schölkopf, A.J. Smola, S. Mika, T. Onoda, and K.-R. Müller, “Robust ensemble learning,” in *Advances in Large Margin Classifiers*, A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, Eds., pp. 207–219. MIT Press, Cambridge, MA, 2000.

[43] R.E. Schapire, “The strength of weak learnability,” *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[44] N. Duffy and D.P. Helmbold, “A geometric approach to leveraging weak learners,” in *Computational Learning Theory: 4th European Conference (EuroCOLT ’99)*, P. Fischer and H. U. Simon, Eds., Mar. 1999, pp. 18–33, Long version to appear in TCS.

[45] A.J. Grove and D. Schuurmans, “Boosting in the limit: Maximizing the margin of learned ensembles,” in *Proceedings of the Fifteenth National Conference on Artifical Intelligence*, 1998.

[46] J. Kivinen and M. Warmuth, “Boosting as entropy projection,” in *Proc. 12th Annu. Conference on Comput. Learning Theory*. 1999, pp. 134–144, ACM Press, New York, NY.

[47] J. Friedman, T. Hastie, and R.J. Tibshirani, “Additive logistic regression: a statistical view of boosting,” *Annals of Statistics*, vol. 2, pp. 337–374, 2000, with discussion pp.375–407, also Technical Report at Department of Statistics, Sequoia Hall, Stanford University.

[48] L. Mason, J. Baxter, P.L. Bartlett, and M. Frean, “Functional gradient techniques for combining hypotheses,” in *Advances in Large Margin Classifiers*, A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, Eds., pp. 221–247. MIT Press, Cambridge, MA, 2000.

[49] G. Rätsch, S. Mika, and M.K. Warmuth, “On the convergence of leveraging,” NeuroCOLT2 Technical Report 98, Royal Holloway College, London, August 2001, A short version is to appear in NIPS 14, MIT Press, 2002.

[50] T. Zhang, “A general greedy approximation algorithm with applications,” in *Advances in Neural Information Processing Systems*. 2002, vol. 14, MIT Press, in press.

[51] L.M. Bregman, “The relaxation method for finding the common point of convex sets and its application to the solution of problems in convex programming,” *USSR Computational Math. and Math. Physics*, vol. 7, pp. 200–127, 1967.

[52] Y. Censor and S.A. Zenios, *Parallel Optimization: Theory, Algorithms and Application*, Numerical Mathematics and Scientific Computation. Oxford University Press, 1997.

[53] M. Collins, R.E. Schapire, and Y. Singer, “Logistic Regression, Adaboost and Bregman distances,” in *Proc. COLT*, San Francisco, 2000, pp. 158–169, Morgan Kaufmann.

[54] S. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, New York, NY, 1996.

[55] G. Rätsch, A. Demiriz, and K. Bennett, “Sparse regression ensembles in infinite and finite hypothesis spaces,” *Machine Learning*, vol. 48, no. 1-3, pp. 193–221, 2002, Special Issue on New Methods for Model Selection and Model Combination. Also NeuroCOLT2 Technical Report NC-TR-2000-085.

[56] R. Cominetti and J.-P. Dussault, “A stable exponential penalty algorithm with superlinear convergence,” *J.O.T.A.*, vol. 83, no. 2, Nov 1994.

[57] M. Doljansky and M. Teboulle, “An interior proximal algorithm and the exponential multiplier method for semidefinite programming,” *SIAM J. Optim.*, vol. 9, no. 1, pp. 1–13, 1998.

[58] L. Mosheyev and M. Zibulevsky, “Penalty/barrier multiplier algorithm for semidefinite programming,” *Optimization Methods and Software*, 1999.

[59] Z.-Q. Luo and P. Tseng, “On the convergence of coordinate descent method for convex differentiable minimization,” *Journal of Optimization Theory and Applications*, vol. 72, no. 1, pp. 7–35, 1992.

[60] S. Della Pietra, V. Della Pietra, and J. Lafferty, “Inducing features of random fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 380–393, Apr. 1997.

[61] T.Poggio and F. Girosi, “Extensions of a theory of networks for approximation and learning: Dimensionality reduction and clustering,” Tech. Rep. AIM-1167, MIT-AILab, Mar. 1990.

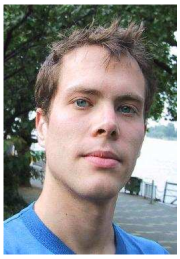
[62] A.J. Smola, *Learning with Kernels*, Ph.D. thesis, Technische Universität Berlin, 1998.

[63] S.L. Salzberg, “A method for identifying splice sites and translational start sites in eukaryotic mRNA,” *Computational Applied Bioscience*, vol. 13, no. 4, pp. 365–376, 1997.

[64] S. Sonnenburg, G. Rätsch, A. Jagota, and K.-R. Müller, “Splice-site recognition with support vector machines,” in preparation, 2002.

[65] P. Hayton, B. Schölkopf, L. Tarassenko, and P. Anuzis, “Support vector novelty detection applied to jet engine vibration spectra,” in *Advances in Neural Information Processing Systems*, T.K. Leen, T.G. Dietterich, and V. Tresp, Eds. 2001, vol. 13, pp. 946–952, MIT Press.

Gunnar Rätsch He received the Diplom in computer science from the University of Potsdam (Germany) in 1998, along with the prize for the best student of the faculty of natural sciences. Three years later, he obtained a Ph.D. in natural sciences with his work on Boosting – being done at the Fraunhofer Research Institute for Computer Architecture and Software Technology in Berlin (Germany). Now he is working as a postdoctoral fellow in the Research School of Information Sciences and Engineering of the Australian National University in Canberra (Australia). His scientific interests and research areas include machine learning, bio-informatics and drug-design.



Sebastian Mika is a doctoral student at Fraunhofer FIRST, Berlin. He received the Diplom in computer science from the Technical University of Berlin in 1998. His scientific interests are in the fields of Machine Learning and kernel methods.

Bernhard Schölkopf received an M.Sc. in mathematics and the Lionel Cooper Memorial Prize from the University of London, followed by the Diplom in physics from the Eberhard-Karls-Universität, Tübingen. In 1997, he obtained a Ph.D. in computer science from the Technical University Berlin. His thesis on Support Vector Learning won the annual dissertation prize of the German Association for Computer Science (GI) 1997. He has worked at AT&T Bell Labs, at GMD FIRST, Berlin, at the Australian National University, at Microsoft Research Cambridge (UK), at Biowulf Technologies (New York), and taught at the Humboldt University and the Technical University Berlin. He is director at the Max-Planck-Institute for Biological Cybernetics. His scientific interests include machine learning and perception.

Klaus-Robert Müller received the Diplom degree in mathematical physics 1989 and the Ph.D. in theoretical computer science in 1992, both from University of Karlsruhe, Germany. From 1992 to 1994 he worked as a Postdoctoral fellow at GMD FIRST, in Berlin where he started to built up the intelligent data analysis (IDA) group. From 1994 to 1995 he was a European Community STP Research Fellow at University of Tokyo in Prof. Amari’s Lab. From 1995 on he is department head of the IDA group at GMD FIRST (since 2001 Fraunhofer FIRST) in Berlin and since 1999 he holds a joint associate Professor position of GMD and University of Potsdam. He has been lecturing at Humboldt University, Technical University Berlin and University of Potsdam. In 1999 he received the annual national prize for pattern recognition (Olympus Prize) awarded by the German pattern recognition society DAGM. He serves in the editorial board of Computational Statistics, IEEE Transactions on Biomedical Engineering and in program and organization committees of various international conferences. His research areas include statistical physics and statistical learning theory for neural networks, support vector machines and ensemble learning techniques. His present interests are expanded to time-series analysis, blind source separation techniques and to statistical denoising methods for the analysis of biomedical data.