Norm-Based Regularization of Boosting

Ying Guo YING.GUO@ANU.EDU.AU
Department of Engineering, FEIT
Australian National University
Canberra 0200, Australia

Peter L. Bartlett PETER.BARTLETT@ANU.EDU.AU
BIOwulf Technologies
Suite 102, 2030 Addison Street
Berkeley, CA 94704, USA

Alexander J. Smola ALEX.SMOLA@ANU.EDU.AU
Computer Sciences Laboratory, RSISE
Australian National University
Canberra, ACT 0200, Australia

Robert C. Williamson BOB.WILLIAMSON@ANU.EDU.AU
Department of Telecommunications Engineering
RSISE, Australian National University
Canberra, ACT 0200, Australia

Jonathan Baxter JBAXTER@WHIZBANG.COM
Whizbang! Labs
4616 Henry Street
Pittsburgh, PA 15213

Editor: U. N. Known (for HTTP://WWW.KERNEL-MACHINES.ORG)

Abstract

AdaBoost has been used with great success as a high-level learning procedure to obtain strong learners from weak learners. It has been shown that this is achieved by gradient descent on a loss function involving the margin of classification. Unfortunately, in the presence of noisy data or an overly complex function class, the algorithm tends to overfit. In this paper we address the problem by adding a regularization term to the overall loss function of AdaBoost. This approach provides a seamless connection between boosting and regularization networks such as support vector machines. Experimental evidence shows the feasibility of our method.

Keywords: Regularization, Boosting, Gradient Descent, Kernels, Entropy Numbers, Support Vector Machines

1. Introduction

In recent years, many researchers have reported significant improvements in the generalization performance using voting methods with learning algorithms such as C4.5 (Quinlan, 1993) or CART (Breiman et al., 1984) as well as with neural networks (Bengio et al., 1994). A number of popular and successful voting methods can be seen as gradient descent

algorithms, which implicitly minimize some loss function of the margin (Breiman, 1999, Friedman et al., 2000, Mason et al., 2000). In particular, the popular AdaBoost algorithm (Freund and Schapire, 1997) can be viewed as a procedure for producing voted classifiers which minimize the sample average of an exponential loss function of the training margins. Although AdaBoost is remarkably successful in practice, AdaBoost's loss function often places too much emphasis on examples with large negative margins. Hence it can suffer from overfitting, particularly in high noise situations.

For the problem of approximating a smooth function from sparse data, regularization techniques (Tikhonov, 1963, Tikhonov and Arsenin, 1977) impose constraints on the approximating set of functions. Rätsch et al. (2001) shows that versions of AdaBoost modified to use regularization are more robust for noisy data. Mason et al. (2000) discusses the problem of approximating a smoother function based on boosting algorithms and a *regularization term* was also suggested to be added to the original loss function.

There is a broad range of choices for the *regularization term*, including many of the popular generalized additive models used in some regularization networks (Girosi et al., 1995). Since the estimates of the target function in boosting algorithms are elements of an inner product space (see next section), one natural way is to construct the *regularization term* as a Hilbert space constraint (Mason et al., 2000, Wahba, 1997).

Since kernel methods have gained popularity recently in machine learning research such as support vector machines (Vapnik, 1995), regularization networks (Girosi et al., 1995), or gaussian processes (Williams, 1998), the *regularization term* can also be designed to be in a reproducing kernel Hilbert space (RKHS) (Aronszajn, 1950).

In this paper, we describe a class of algorithms (which we call NormBoost algorithms) which minimize a regularized risk functional that includes an *error term* and a *regularization term*. Here the *error term* may be, for instance, the original AdaBoost loss function, and the *regularization term* is a function either defined in the Hilbert space of Mason et al. (2000) or a RKHS. Like AdaBoost, NormBoost performs gradient descent so as to maximally reduce the regularized risk functional at each iteration. At the same time, due to the regularization term, NormBoost approximates a smoother function than AdaBoost. That is, instead of only considering the example data, it also takes account of the capacity of the classifier. We show that NormBoost achieves better generalization performance than AdaBoost in noisy situations.

A particular advantage of the RKHS based regularization is that we can appeal to the "representer theorem" (see Theorem 1) which allows us to conclude that the optimal linear combination of weak hypotheses can actually be written using only m terms (where m is the number of examples). The significance of this is that it allows us to perform gradient descent in an m-dimensional space rather than effectively in an infinite dimensional one, as for the standard AdaBoost algorithm (for example). One would expect that this leads to faster convergence, and indeed we demonstrate a marked improvement in convergence rate.

The remainder of the paper is organized as follows. We start by introducing notation and definitions. This includes results from RKHS theory. In Section 3, we introduce the idea of gradient descent, and introduce class of NormBoost algorithms for different regularization terms. Each algorithm chooses linear combinations of elements of an inner product space so as to minimize different regularized risk functions.

In Section 4, we present experimental results for the NormBoost algorithms. These experiments show that the new algorithms based on RKHS norms progressively outperform AdaBoost when the label noise increases. A discussion of the experimental results is included in Section 5.

In Section 6, we use the theoretical results due to (Schapire et al., 1998) to bound the error of a combination of classifiers in terms of the norm bound of the combined classifier.

2. Basic Idea

2.1 Supervised Learning

In supervised learning, the learner is given a set of examples and each example shows what output will be returned for a given input. Here we make the standard assumptions about supervised learning for classification. This means that we assume that a sample of size m, $S := \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ of patterns $\mathbf{x}_i \in X$ and target values $y_i \in \{-1, 1\}$ has been obtained by iid (independent and identically distributed) sampling from a joint probability distribution $\mathrm{P}(\mathbf{x}, y)$. Our goal is to find a function $f : X \to \{-1, 1\}$ such that the value of f corresponds to the most likely label y at location $\mathbf{x}$. In other words, we want to minimize $\mathbf{E}_{(\mathbf{x},y)}\left[\frac{1}{2}(1 - yf(\mathbf{x}))\right]$. We will sometimes refer to the latter as the expected risk of g with respect to $\mathrm{P}(\mathbf{x}, y)$. In this paper, we use the shorthand $\mathbf{X}^m = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$, $Y^m = (y_1, \ldots, y_m)$, $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m))$, and $S^m = (\mathbf{X}^m, Y^m)$.

Our primary interest in this paper is to give a hypothesis f which is a voted combination of classifiers of the form $\mathrm{sign} f(\mathbf{x})$, with

$$f : X \to \mathbb{R}, \text{ where } f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}), \tag{1}$$

where $\alpha_t \in \mathbb{R}$ are the classifier weights, f_t are base classifiers from some class $\mathcal{F}$ and T is the number of base classifiers chosen from $\mathcal{F}$.

Given the above model, we wish to construct a function f such that the *expected error* $\mathbf{E}_{(\mathbf{x},y)}\left[\frac{1}{2}(1 - y\mathrm{sign} f(\mathbf{x}))\right]$, the quantity that we are ultimately interested in, is small. Since we cannot address the problem directly (we do not know $\mathrm{P}(\mathbf{x}, y)$ but only S^m) we have to resort to approximations. Such approximation is usually measured in terms of a *loss function* stating how *bad* it is if the predicted f does not match the true outputs. This function can be designed in terms of the *margin* of S^m. In this paper, the *margin* of an example $(\mathbf{x}_i, y_i)$ with respect to the hypothesis (1) is defined as $yf(\mathbf{x})$. From this definition, we can see that an example is classified correctly if and only if it has a positive margin, and a larger margin can be viewed as a confident correct classifications. So the margin is an indication of the confidence of correct classification.

2.2 The Empirical Risk Functional and the Loss Function

The training of a learning machine commonly involves an *empirical risk functional* that contains a term measuring the *loss* incurred for the training patterns. We use $R_{\mathrm{emp}}[f, S^m]$ to denote the empirical risk functional. It is defined to be the measured mean error rate

with respect to a *loss function* c and for a particular training set S^m,

$$R_{\text{emp}}[f, S^m] = \frac{1}{m} \sum_{i=1}^{m} c(y_i, f(\mathbf{x}_i)). \tag{2}$$

Here we define the *loss function* as $c : \{-1, 1\} \times \mathbb{R} \to \mathbb{R}$. For a specific *loss function* such as $c(y, f(\mathbf{x})) = \frac{1}{2}(y - f(\mathbf{x}))^2$ we obtain

$$R_{\text{emp}}[f, S^m] = \frac{1}{2m} \sum_{i=1}^{m} (y_i - f(\mathbf{x}_i))^2, \tag{3}$$

which forms the least squares method. Next we display some suitable loss functions and their derivatives which will be used in the algorithms.

AdaBoost Loss Function The loss function of AdaBoost is

$$c(y_i, f(\mathbf{x}_i)) = \exp(-y_i f(\mathbf{x}_i)). \tag{4}$$

The derivative of the AdaBoost loss function is

$$c'(y_i, f(\mathbf{x}_i)) = -y_i \exp(-y_i f(\mathbf{x}_i)). \tag{5}$$

Soft Margin Loss Function The soft margin loss function, as introduced by Bennett and Mangasarian (1992), is defined as

$$c(y_i, f(\mathbf{x}_i)) = \max(0, 1 - y_i f(\mathbf{x}_i)) = \begin{cases} 0 & \text{if } y_i f(\mathbf{x}_i) \geq 1 \\ 1 - y_i f(\mathbf{x}_i) & \text{otherwise .} \end{cases} \tag{6}$$

The derivative of the soft margin loss function is

$$c'(y_i, f(\mathbf{x}_i)) = \begin{cases} 0 & \text{if } y_i f(\mathbf{x}_i) \geq 1 \\ -1 & \text{otherwise .} \end{cases} \tag{7}$$

Logistic Loss Function The logistic loss function (cf. (Huber, 1981)) is defined as

$$c(y_i, f(\mathbf{x}_i)) = \ln(1 + \exp(-y_i f(\mathbf{x}_i))). \tag{8}$$

The derivative of the logistic loss function is

$$c'(y_i, f(\mathbf{x}_i)) = -\frac{y_i \exp(-y_i f(\mathbf{x}_i))}{1 + \exp(-y_i f(\mathbf{x}_i))}. \tag{9}$$

Squared Loss Function The squared loss function is just the loss function used in (3),

$$c(y_i, f(\mathbf{x}_i)) = \frac{1}{2}(y_i - f(\mathbf{x}_i))^2. \tag{10}$$

The derivative of the squared loss function (10) is

$$c'(y_i, f(\mathbf{x}_i)) = -(y_i - f(\mathbf{x}_i)). \tag{11}$$

The treatment of each of these loss functions in the framework we present is similar.

2.3 Boosting and AdaBoost

Boosting is a general method for improving the accuracy of learning algorithm. Schapire (1990), Kearns and Valiant (1994) were the first to pose the question of whether a *weak* learning algorithm which performs just slightly better than random guessing can be *boosted* into an arbitrarily accurate *strong* learning algorithm. The AdaBoost algorithm, introduced by Freund and Schapire (1995), solved many of the practical difficulties of the earlier boosting algorithms.

Mason et al. (2000) describe a class of algorithms (called AnyBoost), which are gradient descent algorithms for choosing linear combinations of elements of an inner product space so as to minimize the sample average of some loss function of the margin. One can show (Breiman, 1999, Frean and Downs, 1998, Friedman et al., 2000, Duffy and Helmbold, 1999, Mason et al., 2000) that AdaBoost is an algorithm which performs a gradient descent optimization of the sample average of the loss function (4) of the margins.

Since our main goal will be to study modifications of $R_{\mathrm{emp}}[f, S^m]$ to the extent of adding regularization and smoothness terms, we will use the standard loss function of AdaBoost (4) in the experiments. Clearly, other loss functions such as the logistic loss (8) are also possible (Wahba, 1990, Williams and Barber, 1998). However, unless otherwise specified, we will limit ourselves to (4).

2.4 A Hilbert Space on $\mathbf{X}^m$

Direct minimization of $R_{\mathrm{emp}}[f, S^m]$ may lead to bad generalization performance. Therefore we add a regularization term to the empirical risk. One possibility is to use the norm of f in a Hilbert space, following Mason et al. (2000). They introduce an inner product space $\ell_2^{\mathbf{X}^m}$, with

$$\langle f, g \rangle_{\ell_2^{\mathbf{X}^m}} := \sum_{i=1}^{m} f(\mathbf{x}_i) g(\mathbf{x}_i). \tag{12}$$

This is well defined for all real functions f, g on X. Note that we only regard the inner product *on the sample* $\mathbf{X}^m$ rather than on the space of all possible observations X.

When minimizing $R_{\mathrm{emp}}[f, S^m]$, the Hilbert space norm

$$\|f\|^2_{\ell_2^{\mathbf{X}^m}} := \langle f, f \rangle_{\ell_2^{\mathbf{X}^m}} = \sum_{i=1}^{m} f^2(\mathbf{x}_i) \tag{13}$$

can be used in a regularization term. This leads to our first regularized risk functional:

$$R_{\mathrm{reg}}[f, S^m, \lambda] := R_{\mathrm{emp}}[f, S^m] + \lambda \Omega(\|f\|_{\ell_2^{\mathbf{X}^m}}) \text{ where } \lambda > 0. \tag{14}$$

Here, Ω is a monotonic increasing function and λ is the *regularization constant*. The aim is to find the function $f \in \mathrm{span}(\mathcal{F})$ such that $R_{reg}[f, S^m, \lambda]$ is minimized. The first term in (14) is enforcing closeness to the sample S^m, and the second smoothness, while λ balances the trade-off between goodness-of-fit (small R_{emp}) and simplicity of the hypothesis (small $\Omega(\|f\|_{\ell_2^{\mathbf{X}^m}})$). A convenient choice for Ω is

$$\Omega(\xi) = \frac{1}{2}\xi^2, \text{ i.e., } \Omega(\|f\|_{\ell_2^{\mathbf{X}^m}}) = \frac{1}{2}\|f\|^2_{\ell_2^{\mathbf{X}^m}}. \tag{15}$$

2.5 Reproducing Kernel Hilbert Spaces

A reproducing kernel Hilbert space (RKHS) (Aronszajn, 1950) is a Hilbert space of functions defined over some domain X with the property that, for each $\mathbf{x} \in X$, the evaluation functionals $\mathrm{V}_{\mathbf{x}}[f]$, defined as

$$\mathrm{V}_{\mathbf{x}}[f] = f(\mathbf{x}), \qquad \forall f \in \mathcal{H}_k$$

are linear, bounded functionals. It can be proved that for every RKHS, we can associate $\mathcal{H}_k$ a positive definite function $k(\mathbf{x}, \mathbf{y})$, which is called the reproducing kernel of $\mathcal{H}_k$. The kernel of $\mathcal{H}_k$ has the following *reproducing property*:

$$f(\mathbf{x}) = \langle f(\cdot), k(\cdot, \mathbf{x}) \rangle_{\mathcal{H}_k}, \qquad \forall f \in \mathcal{H}_k, \qquad \forall \mathbf{x} \in X,$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}_k}$ denotes the inner product in $\mathcal{H}_k$, i.e. k is the *representer of evaluation*. Using the reproducing property of the kernel k we have

$$\langle k(\cdot, \mathbf{x}_i), k(\cdot, \mathbf{x}_j) \rangle_{\mathcal{H}_k} = k(\mathbf{x}_i, \mathbf{x}_j). \tag{16}$$

By using a RKHS, we can design our second regularized risk functional

$$R_{reg}[f] = R_{\mathrm{emp}}[f] + \lambda \Omega \left(\|f\|_{\mathcal{H}_k} \right) \text{ where } \lambda > 0. \tag{17}$$

Here, Ω is a monotonic increasing function, $\| \cdot \|_{\mathcal{H}_k}$ denotes the corresponding RKHS norm, $\| \cdot \|_{\mathcal{H}_k} = \sqrt{\langle \cdot, \cdot \rangle_{\mathcal{H}_k}}$, and λ is the *regularization constant* as in (14).

2.6 Representer Theorem

Kimeldorf and Wahba (1971) and later Cox and O'Sullivan (1990) proved that for R_{emp} of the form (2) (which measures the difference between the $f(\mathbf{x}_i)$ and y_i), the minimizer of (17) can be written as

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, \mathbf{x}_i), \tag{18}$$

where $k : X \times X \to Y$ is the kernel of the corresponding $\mathcal{H}_k$. On S^m this implies $\mathbf{f} = K\boldsymbol{\alpha}$, where the matrix $K_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$, the weight vector $\boldsymbol{\alpha}^T := (\alpha_1, \alpha_2, \ldots, \alpha_m)$, $\mathbf{f} = (f(\mathbf{x}_1), \cdots, f(\mathbf{x}_m))$ as defined before. This result is known as the Representer Theorem (Kimeldorf and Wahba, 1971).

More generally, for an arbitrary loss function, the following theorem was proved by Schölkopf et al. (2000).

Theorem 1 (Nonparametric Representer Theorem) *Suppose we are given a RKHS $\mathcal{H}$ with the positive definite real-valued kernel $k : X \times X \to \mathbb{R}$, a training set S^m, a strictly monotonic increasing real-valued function Ω on $[0, \infty]$, an arbitrary loss function c: $(X \times \mathbb{R})^m \times \mathbb{R}^m \to \mathbb{R}^+ \cup \{\infty\}$, a class of functions*

$$\mathcal{F} = \left\{ f \in \mathbb{R}^X \;\middle|\; f(\cdot) = \sum_{i=1}^{\infty} \beta_i k(\cdot, \mathbf{x}_i), \beta_i \in \mathbb{R}, \mathbf{x}_i \in X, \|f\|_{\mathcal{H}_k} < \infty \right\},$$

and $\bar{\mathcal{F}}$ the closure of $\mathcal{F}$ in $\mathcal{H}$. Then any $f \in \bar{\mathcal{F}}$ minimizing the regularized risk functional

$$c(S^m, f(\mathbf{x}_1), \cdots, f(\mathbf{x}_m)) + \Omega(\|f\|_{\mathcal{H}_k}) \tag{19}$$

admits a representation of the form of (18).

This theorem shows that for a large class of algorithms minimizing a sum of an empirical risk term and a regularization term in an RKHS, the optimal solutions can be written as expansions in terms of training examples. The optimal function f_{opt} in $\mathcal{F}$ minimizing (17) can be written using only m terms. For such an f_{opt}, we have

$$\|f_{opt}\|_{\mathcal{H}}^2 = \langle f_{opt}, f_{opt}\rangle_{\mathcal{H}} = \left\langle \sum_{i=1}^{m} \alpha_i k(\cdot, \mathbf{x}_i), \sum_{j=1}^{m} \alpha_j k(\cdot, \mathbf{x}_j) \right\rangle_{\mathcal{H}} = \sum_{i,j=1}^{m} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}. \tag{20}$$

Thus rather than dealing with all functions in the (possibly infinite dimensional) Hilbert space $\mathcal{H}$ as in traditional boosting algorithms, all one has to do for RKHS algorithms is to find a finite set of parameters α_i $(i = 1, \ldots, m)$. Since the optimization is within a lower dimensional space, we expect that the algorithm will converge much faster than standard boosting algorithms.

3. Gradient Descent

3.1 Basic Idea

This leads to the practical question of how minimizers of the regularized risk functional can be obtained efficiently. One possibility to minimize (2) is gradient descent. Mason et al. (2000) propose such an algorithm in function space.

This involves the computation of the gradient of $R_{\text{reg}}[f, S^m, \lambda]$ with respect to f, represented in *some* space, and taking a descent step in the negative gradient direction to minimize the objective function $R_{\text{reg}}[f, S^m, \lambda]$. For instance the gradient could be computed with respect to the space of the coefficients $(\alpha_1, \ldots, \alpha_T)$ or likewise with respect to $\ell_2^{\mathbf{X}^m}$ which was introduced via the dot product (12). Further choices, such as a reproducing kernel Hilbert space are possible and much of the paper will focus on the distinctions between the different spaces and their practical performance.

We use the gradient descent algorithm (Algorithm 1) to minimize the regularized risk functional $R_{\text{reg}}[f, S^m, \lambda]$. At each step we compute the gradient with respect to f or one of its parametric representations via $\boldsymbol{\alpha}$, take a step in the negative direction of the gradient, and repeat the procedure until convergence.

If we perform descent in the space of $\boldsymbol{\alpha}$ rather than f, we have to replace the derivatives in f by the corresponding terms in $\boldsymbol{\alpha}$ and modify the update equations in f accordingly.

Since a unit step in the direction of the negative gradient of $R_{reg}[f]$ does not necessarily guarantee that $R_{reg}[f]$ will decrease, it is advantageous in many cases to perform a line-search in the direction of $\nabla R_{reg}[f]$, i.e. seek the step size Λ such that $R_{reg}\left[f - \Lambda \nabla R_{reg}[f]\right]$ is minimized. One can obtain the approximately optimal step size Λ by, for instance, starting with an initial guess of an interval, and if no minimum can be found strictly inside the interval, doubling its size. (see, for example, (Schölkopf and Smola, 2001) chapter 6.)

Algorithm 1 Gradient descent algorithm

input Training data S^m, loss function c, regularizer $\Omega(\cdot)$, inner product $\langle\cdot,\cdot\rangle$, tolerance ε, step length Λ (unless line-search selected)
  repeat
    Compute $g := \nabla_f R_{\text{reg}}[f, S^m, \lambda]$.
    if linesearch **then**
      $\Lambda = \arg\min_\gamma R_{\text{reg}}[f - \gamma g, S^m, \lambda]$.
    end if
    $f \to f - \Lambda g$.
  until $\|\nabla_f R_{\text{reg}}[f, S^m, \lambda]\| \leq \varepsilon$

3.2 A Lower Bound on $R_{\text{reg}}[f, S^m, \lambda]$

Before we proceed with the actual implementation details we consider a suitable stopping rule. The following theorem shows that the size of the gradient is a good criterion to assess the quality of the current solution.

Theorem 2 (Lower Bound on Primal Objective Function) *Denote by $R_{\text{emp}}[f]$ a convex and differentiable functional on a Hilbert space H and consider the regularized risk functional*

$$R_{\text{reg}}[f] = R_{\text{emp}}[f] + \lambda\Omega(\|f\|) \text{ where } \lambda > 0 \tag{21}$$

Then for any $f, \Delta f \in H$, and for $\Omega(\|f\|) = \frac{1}{2}\|f\|^2$,

$$R_{\text{reg}}[f] - R_{\text{reg}}[f - \Delta f] \quad \leq \quad \frac{1}{2\lambda}\|\nabla R_{\text{reg}}[f]\|^2. \tag{22}$$

Proof We start the proof in a more general way which will be useful later. We assume that $f - \Delta f$ is the minimizer of $R_{\text{reg}}[f]$ since proving the inequality for the minimizer is sufficient. Since R_{emp} is convex and differentiable we know that

$$R_{\text{emp}}[f] - R_{\text{emp}}[f - \Delta f] \leq \langle \Delta f, \nabla R_{\text{emp}}[f]\rangle. \tag{23}$$

Therefore we may bound $\rho(f, \Delta f) := R_{\text{reg}}[f] - R_{\text{reg}}[f - \Delta f]$ by

$$\begin{aligned} \rho(f, \Delta f) &= R_{\text{emp}}[f] - R_{\text{emp}}[f - \Delta f] + \lambda\Omega(\|f\|) - \lambda\Omega(\|f - \Delta f\|) \\ &\leq \langle \Delta f, \nabla R_{\text{emp}}[f]\rangle + \lambda(\Omega(\|f\|) - \Omega(\|f - \Delta f\|)). \end{aligned} \tag{24}$$

This is in particular true for the minimization of (24) with respect to Δf. This minimum satisfies $\nabla_{\Delta f}\rho(f, \Delta f) = 0$ and therefore

$$\begin{aligned} &\nabla R_{\text{emp}}[f] - \lambda\nabla_{\Delta f}\Omega(\|f - \Delta f\|) \\ &= \nabla R_{\text{emp}}[f] + \lambda\Omega'(\|f - \Delta f\|)\frac{f - \Delta f}{\|f - \Delta f\|} = 0. \end{aligned} \tag{25}$$

Here we used

$$\begin{aligned} \nabla_{\Delta f}(\|f - \Delta f\|) &= \nabla_{\Delta f}\langle f - \Delta f, f - \Delta f\rangle^{\frac{1}{2}} \\ &= \frac{1}{2}(\|f - \Delta f\|)^{-1}\nabla_{\Delta f}\langle f - \Delta f, f - \Delta f\rangle = -\frac{f - \Delta f}{\|f - \Delta f\|}. \end{aligned}$$

For the special case where $\Omega(\|f\|) = \frac{1}{2}\|f\|^2$, we get $\Omega'(\|f - \Delta f\|) = \|f - \Delta f\|$. From (25) we have

$$\nabla R_{\text{emp}}[f] + \lambda \|f - \Delta f\| \frac{f - \Delta f}{\|f - \Delta f\|} = 0.$$

So (24) is minimized by

$$\Delta f = \nabla R_{\text{emp}}[f]/\lambda + f.$$

Substituting this back into the upper bound (24) on $\rho(f, \Delta f)$, also noticing that when $\Omega(\|f\|) = \frac{1}{2}\|f\|^2$,

$$\nabla R_{\text{reg}}[f] = \nabla R_{\text{emp}}[f] + \lambda f,$$

we have

$$\begin{aligned}\rho(f, \Delta f) &\leq \langle \nabla R_{\text{emp}}[f]/\lambda + f, R_{\text{emp}}[f] \rangle + \frac{\lambda}{2}\left(\|f\|^2 - \|\frac{R_{\text{emp}}[f]}{\lambda}\|^2 \right) \\ &= \frac{1}{2\lambda}\left(\|R_{\text{emp}}[f]\|^2 + 2\lambda \langle f, \nabla R_{\text{emp}}[f] \rangle + \lambda^2 \|f\|^2 \right) = \frac{1}{2\lambda}\|\nabla R_{\text{reg}}[f]\|^2.\end{aligned}$$

■

For general Ω, matters are somewhat more delicate. We can rewrite (25) as

$$\lambda \Omega'(\|h\|) \frac{h}{\|h\|} = -\nabla R_{\text{emp}}[f], \text{ where } h := f - \Delta f. \tag{26}$$

This equation can be solved for

$$h = -\omega \frac{\nabla R_{\text{emp}}[f]}{\|\nabla R_{\text{emp}}[f]\|}, \text{ for some } \omega > 0. \tag{27}$$

This leads to

$$\lambda \Omega'(\omega) = \zeta, \text{ where } \zeta = \|\nabla R_{\text{emp}}[f]\|.$$

If the function $\Omega' : \mathbb{R}^+ \to \mathbb{R}$ is invertible, the equation $\Omega'(\omega) = \zeta/\lambda$ can be solved for ω and we obtain $\omega = (\Omega')^{-1}(\zeta/\lambda)$. Substituting ω back into (27), we achieve

$$f - \Delta f = h = -(\Omega')^{-1}\left(\frac{\zeta}{\lambda}\right) \frac{\nabla R_{\text{emp}}[f]}{\|\nabla R_{\text{emp}}[f]\|}.$$

This value of $f - \Delta f$ can then be used in a lower bound for $R_{\text{emp}}[f]$. At the same time, it also offers a cheap replacement for the line search strategy in the gradient descent algorithm, especially if the evaluation of $R_{\text{emp}}[f]$ is expensive in comparison to the other terms of the regularized risk functional. Finally note that these bounds are tight if the linearization of $R_{\text{emp}}[f]$ is exact. Therefore it makes an excellent stopping criterion.

3.3 Gradient Descent in Function Space

Let us start with gradients in function space $\mathcal{F}$. Here the regularized risk functionals are given by (14) and (17). There is a wide choice for the regularization operator Ω. In the

following gradient descent calculation and experiments, we choose $\Omega(\xi) = \frac{1}{2}\xi^2$. Hence we have

$$R_{reg}[f] = \frac{1}{m}\sum_{i=1}^{m} c(y_i, f(\mathbf{x}_i)) + \frac{1}{2}\xi^2, \tag{28}$$

We define

$$\beta_i := \frac{1}{m}c'(y_i, f(\mathbf{x}_i)), \qquad i = 1, \ldots, m, \tag{29}$$

and the vector

$$\boldsymbol{\beta}^T = (\beta_1, \beta_2, \ldots, \beta_m). \tag{30}$$

Thus the vector $\boldsymbol{\beta}$ can be computed from the loss function.

3.3.1 $\ell_2^{\mathbf{X}^m}$ NormBoost

Consider the regularized risk functional (14) first. It can be expressed as

$$R_{reg}[f] = \frac{1}{m}\sum_{i=1}^{m} c(y_i, f(\mathbf{x}_i)) + \frac{\lambda}{2}\|f\|^2_{\ell_2^{\mathbf{X}^m}}, \tag{31}$$

where we still use (4) as the loss function. In this case, the gradient of the loss function is

$$\nabla_{\mathbf{f}} R_{\text{emp}}[f](\mathbf{x}) = \boldsymbol{\beta}, \tag{32}$$

where $\mathbf{f}$ is the vector of the value of f on $\mathbf{X}^m$ as defined before. Equation (31) can be rewritten as:

$$\nabla_{\mathbf{f}} R_{\text{reg}}[f](\mathbf{x}) = \boldsymbol{\beta} + \lambda f,$$

and we obtain the following update rules for f, given a step size Λ.

$$f_{t+1} = f_t - \Lambda(\boldsymbol{\beta} + \lambda f). \tag{33}$$

For computational reasons we *represent* f as a linear combination of functions in a finite dimensional space. With the expansion of $f(\mathbf{x})$ as in (1) the update rule for the weights becomes

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t - \Lambda(\mathbf{F}^{-1}\boldsymbol{\beta} + \lambda\boldsymbol{\alpha}_t), \tag{34}$$

where the matrix $\mathbf{F}_{ij} := f_j(\mathbf{x}_i)$. The inverse of $\mathbf{F}$ is required in (34). $\mathbf{F}$ may not be invertible, and it is also expensive to compute $\mathbf{F}^{-1}\boldsymbol{\beta}$, hence we do not use the gradient descent in function space for $\ell_2^{\mathbf{X}^m}$ norm regularization.

3.3.2 RKHS NormBoost

Next we consider the regularized risk functional (17) with the regularizer term as in (28). The decision function is defined by (18). Combining with (2), the gradient of the regularized risk functional can be expressed as

$$\begin{aligned}\nabla_f R_{reg}[f] &= \nabla_f \left(\frac{1}{m}\sum_{i=1}^{m} c(y_i, f(\mathbf{x}_i)) + \frac{\lambda}{2}\|f\|_{\mathcal{H}_k}\right) \\ &= \sum_{i=1}^{m}\frac{1}{m}c'(y_i, f(\mathbf{x}_i))k(\cdot, \mathbf{x}_i) + \lambda f = \sum_{i=1}^{m}\beta_i k(\cdot, \mathbf{x}_i) + \lambda f.\end{aligned} \tag{35}$$

We obtain the update rules for f:

$$\begin{aligned} f_{t+1} &= f_t - \Lambda \left(\sum_{i=1}^{m} \beta_i k(\cdot, \mathbf{x}_i) + \lambda f \right) \\ &= \sum_{i=1}^{m} \alpha_i k(\cdot, \mathbf{x}_i) - \Lambda \left(\sum_{i=1}^{m} \beta_i k(\cdot, \mathbf{x}_i) + \lambda \sum_{i=1}^{m} \alpha_i k(\cdot, \mathbf{x}_i) \right). \end{aligned} \tag{36}$$

Hence the update rule for the weights becomes

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t - \Lambda(\boldsymbol{\beta} + \lambda \boldsymbol{\alpha}_t). \tag{37}$$

3.4 Gradient Descent in Coefficient Space

For comparison, consider the corresponding updates for gradient descent in *coefficient space*. We will describe the update rules for $\ell_2^{\mathbf{X}^m}$ norm regularization and RKHS norm regularization separately. In order to compare easily, we use the same regularized risk functional, loss function, and function space as before.

3.4.1 $\ell_2^{\mathbf{X}^m}$ NormBoost

In the $\ell_2^{\mathbf{X}^m}$ norm regularization as above, the gradient descent of the regularization functional is

$$\begin{aligned} \nabla_{\boldsymbol{\alpha}} R_{\mathrm{reg}}[f] &= \nabla_{\boldsymbol{\alpha}} \left(\frac{1}{m} \sum_{i=1}^{m} c(y_i, f(\mathbf{x}_i)) + \frac{\lambda}{2} \|f\|^2_{\ell_2^{\mathbf{X}^m}} \right) \\ &= \mathbf{F}^T \boldsymbol{\beta} + \lambda \mathbf{F}^T \mathbf{F} \boldsymbol{\alpha}. \end{aligned} \tag{38}$$

Hence the update rule for the weights is

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t - \Lambda \mathbf{F}^T (\boldsymbol{\beta} + \lambda \mathbf{F} \boldsymbol{\alpha}_t) \tag{39}$$

3.4.2 RKHS NormBoost

It is straightforward to see that in the case of RKHS Norm regularization as above, we obtain

$$\nabla_{\boldsymbol{\alpha}} R_{reg}[f] = K\boldsymbol{\beta} + \lambda K \boldsymbol{\alpha} \tag{40}$$

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t - \Lambda K \boldsymbol{\beta} - \Lambda \lambda K \boldsymbol{\alpha}_t. \tag{41}$$

In other words, the updates from (34) are multiplied by the kernel matrix K to obtain the update rules in the coefficient space. This means that we are performing gradient descent in a space with respect to the metric given by K rather than the Euclidean metric.

3.4.3 Other regularization operators

Beyond the function space based regularization functions, we may also use regularization directly in coefficient space. The family of ℓ_p norms is particularly useful here. Next we show two examples in this case.

Regularization	$\frac{1}{2}\|f\|^2_{\mathcal{H}}$	$\|f\|_{\mathcal{H}}$	$\frac{1}{2}\|f\|^2_{\ell_2^{X^m}}$	$\|f\|_{\ell_2^{X^m}}$	$\frac{1}{2}\|\boldsymbol{\alpha}\|^2_{\ell_2}$	$\|\boldsymbol{\alpha}\|_{\ell_1}$
Gradient in f	$\boldsymbol{\alpha}$	$\frac{\boldsymbol{\alpha}}{\sqrt{\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}}}$	$K\boldsymbol{\alpha}$	$\frac{K\boldsymbol{\alpha}}{\sqrt{\boldsymbol{\alpha}^\top K^\top K\boldsymbol{\alpha}}}$	—	—
Gradient in $\boldsymbol{\alpha}$	$K\boldsymbol{\alpha}$	$\frac{K\boldsymbol{\alpha}}{\sqrt{\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}}}$	$K^\top K\boldsymbol{\alpha}$	$\frac{K^\top K\boldsymbol{\alpha}}{\sqrt{\boldsymbol{\alpha}^\top K^\top K\boldsymbol{\alpha}}}$	$\boldsymbol{\alpha}$	$\mathrm{sign}(\boldsymbol{\alpha})$

Table 1: Different regularization functionals and their gradients in function space and coefficient space

For $\Omega(f) := \frac{1}{2}\|\boldsymbol{\alpha}\|^2_{\ell_2}$, the gradient of the regularized risk functional is

$$\nabla_{\boldsymbol{\alpha}} R_{\mathrm{reg}}[f] = K\boldsymbol{\beta} + \lambda\boldsymbol{\alpha}. \tag{42}$$

So the update rule is

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t - \Lambda(K\boldsymbol{\beta} + \lambda\boldsymbol{\alpha}).$$

For $\Omega(f) := \|\boldsymbol{\alpha}\|_{\ell_1}$, the gradient of the regularized risk functional is

$$\nabla_{\boldsymbol{\alpha}} R_{\mathrm{reg}}[f] = K\boldsymbol{\beta} + \lambda\mathrm{sign}(\boldsymbol{\alpha}). \tag{43}$$

So the update rule is

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t - \Lambda(K\boldsymbol{\beta} + \lambda\mathrm{sign}(\boldsymbol{\alpha})).$$

Table 1 gives an overview of different regularization operators and their gradients in the RKHS inner product space.

4. Experiments

In order to compare the performance of NormBoost and AdaBoost, we study the performance of both algorithms on three artificial datasets. The main points we want to assess are the classification accuracy and the convergence speed. We use $R[f, P]$ as the *expected error* of a combined classifier f with respect to the probability distribution P on $Z = X \times \{-1, 1\}$, and $R_{est}[f, P]$ as the *estimate error* of $R[f, P]$ which is the approximation of $R[f, P]$ from the training set S^m.

Dataset 1 The first dataset was generated in a two dimensional square $L = \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\|_{\ell_\infty} \leq 1\}$. The true classification function to be learned was 1 when the points were in $L_{positive} = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 \geq x_2\}$, and -1 otherwise. For each trial a training sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{50}$ of size 50 was generated with the $\mathbf{x}_i$ drawn independently at random from a uniform distribution on L.

Dataset 2 The second dataset was generated by drawing d dimensional points uniformly from $[-1, 1]^d$, and using them as inputs to a map defined by a mixture of n Gaussian bumps with width chosen from a uniform distribution on the interval $[0.1, 1]$ and weights chosen from a uniform distribution on the interval $[0.0, 1.0]$. Patterns were split into two classes.

Dataset 3 The third dataset is Banana, which is available at *www.first.gmd.de/∼raetsch/*.

For both the first and second datasets, independent label noise of intensity 5%, 10%, 15% and 20% was applied to the labels y_i. Once a hypothesis f was found by the procedure, an estimate error $R_{est}(f, P)$ of the expected error $R(f, P)$ was obtained via 10000 independently drawn test points. All of the experiments were repeated 10 times with the examples randomly selected for training and test purposes. The results were then averaged over the 10 repeats.

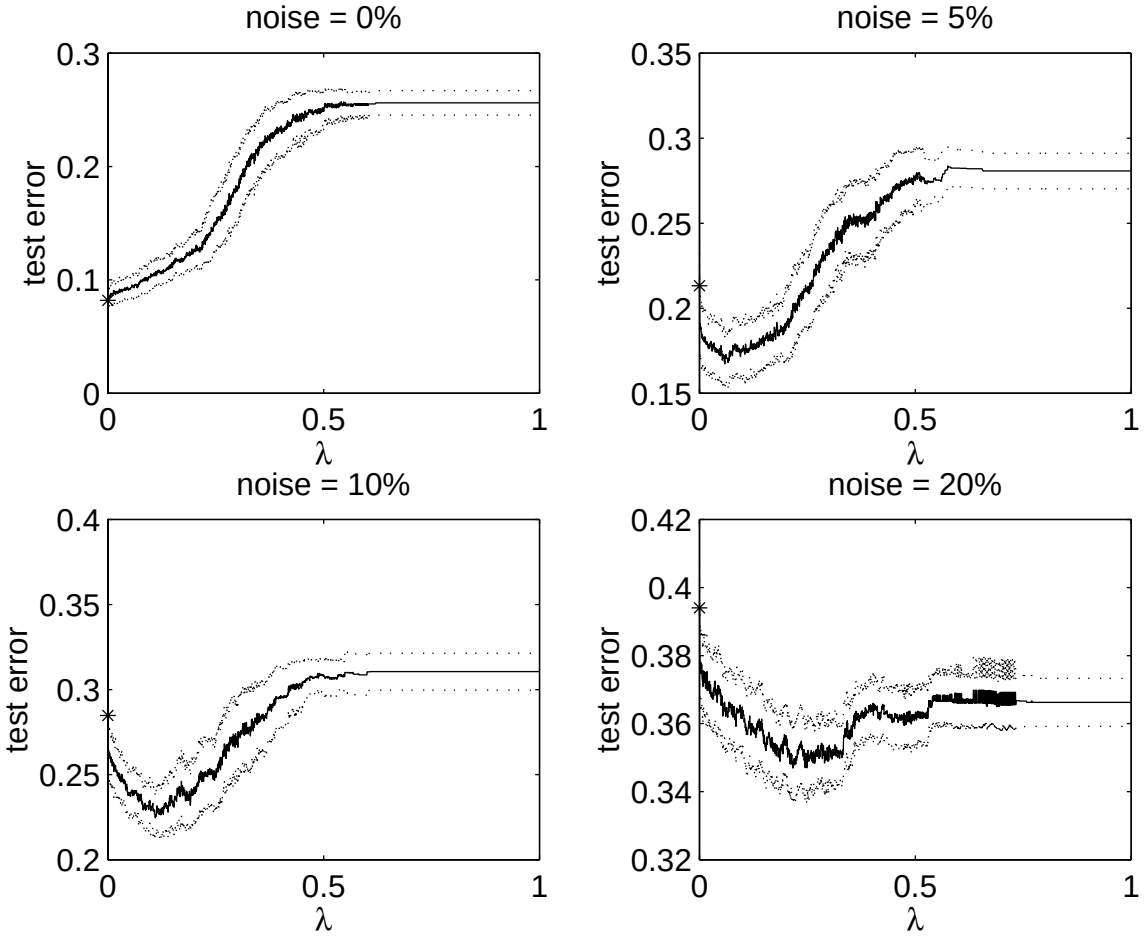

Figure 1: Test error versus λ for $\ell_2^{\mathbf{X}^m}$ NormBoost Algorithm (the short-dashed curves are ± 1 standard deviation).

4.1 $\ell_2^{\mathbf{X}^m}$ NormBoost and AdaBoost

For all the experiments in $\ell_2^{\mathbf{X}^m}$ NormBoost, axis-orthogonal hyperplanes (also called decision stumps) were produced by the weak learner.

Figure 1 (first dataset) shows the test error $R_{est}(f, P)$ versus the regularization parameter λ with different noise levels. The $\lambda = 0$ points show the test error of AdaBoost. With

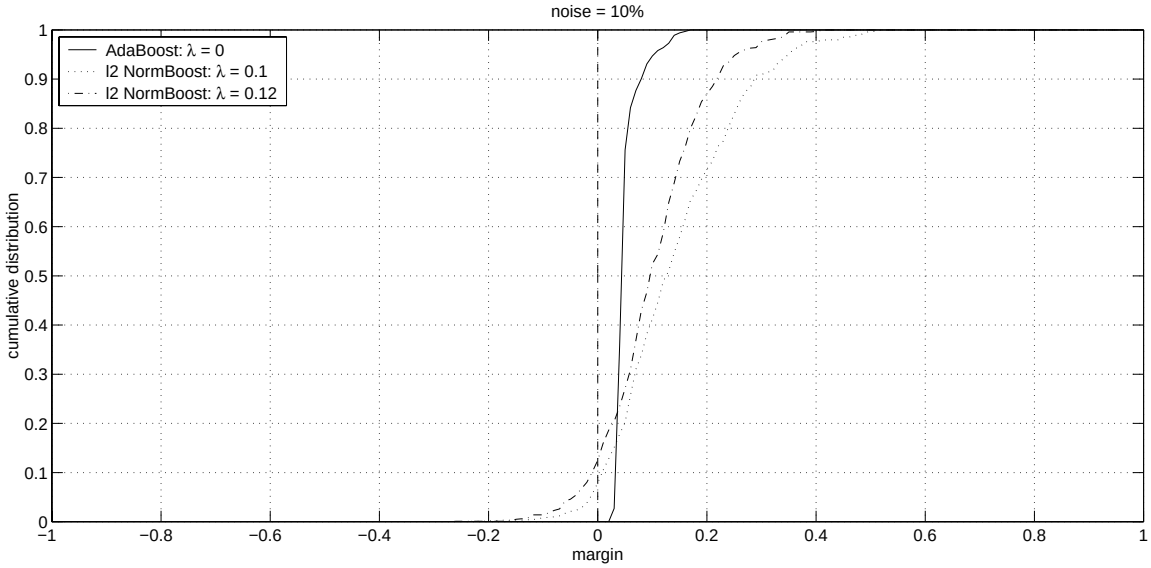

Figure 2: The margin distribution for AdaBoost and NormBoost on noise training sets (see text).

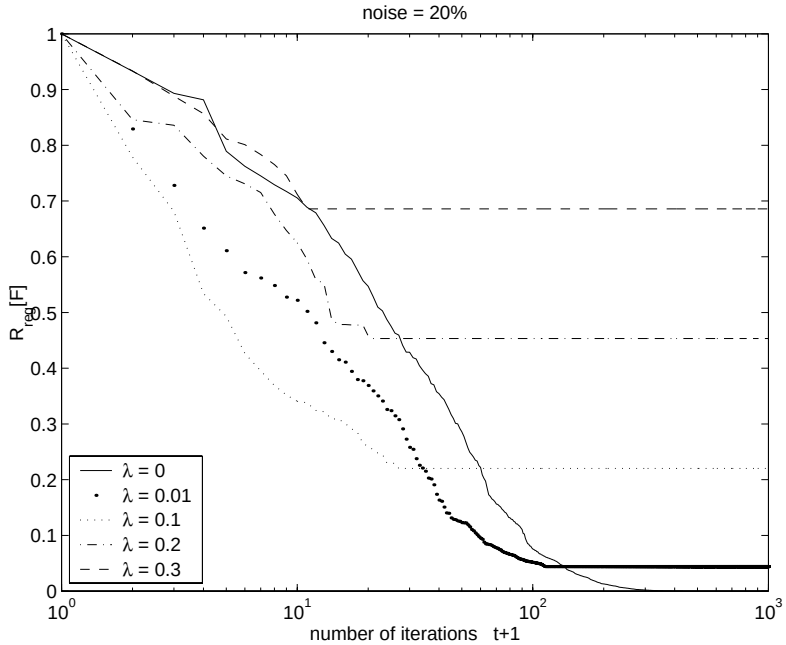

Figure 3: R_{reg} versus the iteration time t for $\ell_2^{\mathbf{X}^m}$ NormBoost Algorithm.

no noise, AdaBoost shows better performance than $\ell_2^{\mathbf{X}^m}$ NormBoost. But when the training sample is noisy, the lowest test error (see Figure 1) is achieved at some $\lambda^* > 0$ instead of the

original points. The stars in Figure 1 symbolize the test error of AdaBoost (with $\lambda = 0$). We notice that there is a jump at $\lambda = 0$ which will be discussed in Section 5.

Margin distribution graphs (Schapire et al., 1998) are useful to analyze the experimental results. We know that a large positive margin can be interpreted as a "confident" correct classification. AdaBoost tends to increase the margins associated with examples and converge to a margin distribution in which most examples have large positive margins. Hence, AdaBoost can improve the generalization error of a classifier when there is no noise. But when there is noise in the training set, AdaBoost generates an overfitting classifier by trying to classify the noisy points with positive margins. In fact, in most cases, the boosting algorithm will modify the training sample distribution to force the learner to concentrate on its errors, and will thereby force the learner to concentrate on learning noisy examples.

The NormBoost algorithm, because of the balancing influence of the *regularization term*, tends to find a smoother classifier and converge to a margin distribution in which some examples may have negative margins.

Figure 2 (first dataset) shows the margin distributions after 10000 iterations for AdaBoost, $\ell_2^{\mathbf{X}^m}$ NormBoost with $\lambda = 0.1$, and $\ell_2^{\mathbf{X}^m}$ NormBoost with $\lambda = 0.12$, indicated by solid, dotted, and dashed lines, respectively. With noise, about 10% of the training data are classified incorrectly, while the margins of more than 75% of the points are bigger than those of AdaBoost.

The convergence of the regularized risk functional versus the iteration time t is shown in Figure 3 (first dataset), where the curve for $\lambda = 0$ depicts the R_{reg} of AdaBoost and the other curves depict that of $\ell_2^{\mathbf{X}^m}$ NormBoost with different λ levels. Our observation is that the regularized risk functionals of $\ell_2^{\mathbf{X}^m}$ NormBoost converge much faster than AdaBoost. Notice that the number of iterations is plotted on a log scale.

4.2 RKHS NormBoost and AdaBoost

For all the experiments on RKHS NormBoost, we chose Gaussian radial basis functions with width σ as the kernel function class, i.e.

$$\mathcal{F}_\sigma = \left\{ \mathbf{x} \mapsto \exp\left(-\frac{1}{\sigma^2} \|\mathbf{x} - \mathbf{x}_i\|^2 \right),\ i = 1, \ldots, m \right\}, \tag{44}$$

and $\Omega\left(f(\boldsymbol{\alpha})\right) := \frac{1}{2}\|f(\boldsymbol{\alpha})\|_{\mathcal{H}_k}^2$.

Figure 4 (second dataset) shows the test error as a function of λ for a range of different kernel widths σ and different noise levels. The regularization parameter λ is plotted on a log scale. As expected, the curves have their lowest point at some $\lambda^* > 0$ for some appropriate values of σ. Hence, there are two parameters that effectively influence the generalization performance of RKHS NormBoost: (λ, σ).

This observation leads us to consider the relationship between RKHS NormBoost and support vector machines. Support vector (SV) machines are linear classifiers that use the maximum margin hyperplane in a feature space also defined by a kernel function. Using the same notation $\mathbf{X}^m$, we can denote the hypothesis class implemented by SV machines on $\mathbf{X}^m$ with weight vector bounded by $R_{\mathbf{w}}$:

$$\mathcal{F}_{R_{\mathbf{w}}} = \left\{ \mathbf{x} \mapsto \sum_{i=1}^m \alpha_i k(\mathbf{x}, \mathbf{x}_i) \colon \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \le R_{\mathbf{w}}^2 \right\}, \tag{45}$$

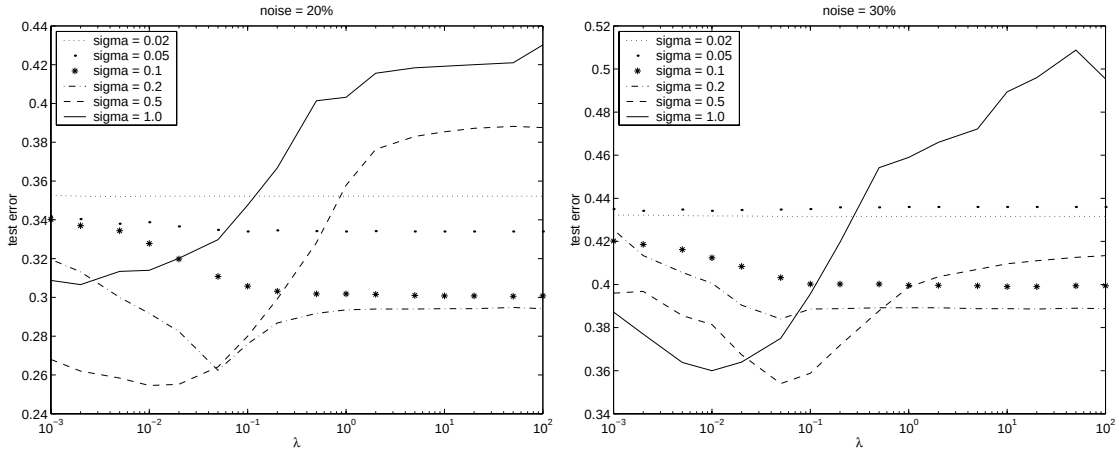

Figure 4: Test error versus λ for RKHS NormBoost Algorithm.

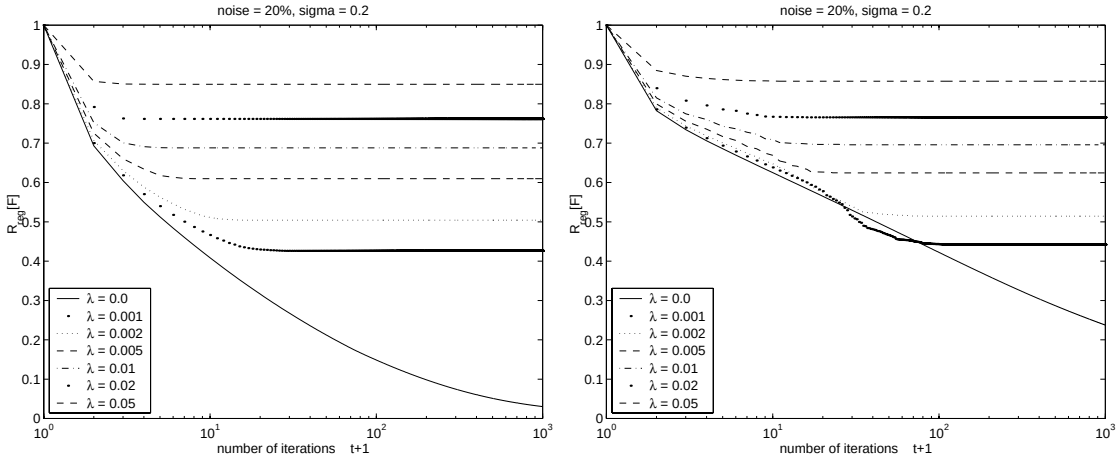

Figure 5: R_{reg} versus the iteration time t for RKHS NormBoost Algorithm. Left: Gradient in Function Space. Right: Gradient in Coefficient Space. Observe that for a noise level of 20%, $\sigma = 0.2$, $\lambda = 0.05$ gave the smallest test error (26%). For these values of σ and λ, the algorithm converged in only two steps.

where $0 \leq \alpha_i \leq C$, for $i = 1, \ldots, m$ and $C \in \mathbb{R}$.

Recalling the output of RKHS NormBoost is $f(\mathbf{x}) = \sum_{t=1}^{m} \alpha_t k(\mathbf{x}, \mathbf{x}_t)$ and

$$\| f \|_{\mathcal{H}_k}^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) < \infty,$$

by the definition of the RKHS, we find that RKHS NormBoost has the same hypothesis class form (45) as SV machines. The two parameters (C, σ) or (ν, σ) (see (Cristianini and Shawe-Taylor, 2000)) play important roles in SV machines. Similarly the parameters (λ, σ) effectively determine the character of an RKHS NormBoost algorithm.

The choice of parameterization will have an impact on the performance of the algorithm. Fixing the noise level (20%) and $\sigma = 0.2$, for different values of λ on our second toy problem, Figure 5 (second dataset) graphs the convergence speed for both experiments: one gradient in the function space and another in the weight space. Notice that $R_{reg}[f]$ converges in less than 100 iterations of RKHS NormBoost while $R_{reg}[f]$ is still decreasing after 1000 iterations for AdaBoost. We also observed that the gradient descent in function space is never slower (in terms of number of iterations) and is usually considerably faster. Furthermore it is computationally less expensive, also on a per-iteration basis.

In Figure 5, different curves correspond to different values of λ from the long-dashed curve in Figure 4 (left). In Figure 5, the long-dashed curve corresponds to the value $\lambda = 0.05$ that achieves the minimum test error in Figure 4. For this "good" choice of λ, RKHS NormBoost converged in only 3 steps when R_{reg} is defined using the gradient in function space and only 10 steps when R_{reg} is defined using the gradient in coefficient space.

The convergence properties of $R_{reg}[f]$ are similar in the second and third datasets; see Figures 6 (second dataset) and 7 (third dataset). Gradient descent in the function space $\mathcal{F}$ with respect to the Euclidean metric is smoother than in the weight space $\boldsymbol{\alpha}$ with respect to the metric given by the kernel matrix K. The performance of the RKHS NormBoost shows up more clearly in Figure 7, where the results for the dataset BANANA are given. The results show that the performance of RKHS NormBoost is comparable to other learning algorithms for suitable choice of λ and σ. In this experiment, the lowest classification error achieved was $10.43\% \pm 0.47\%$, corresponding to $\lambda = 3$ and $\sigma = 0.6$.

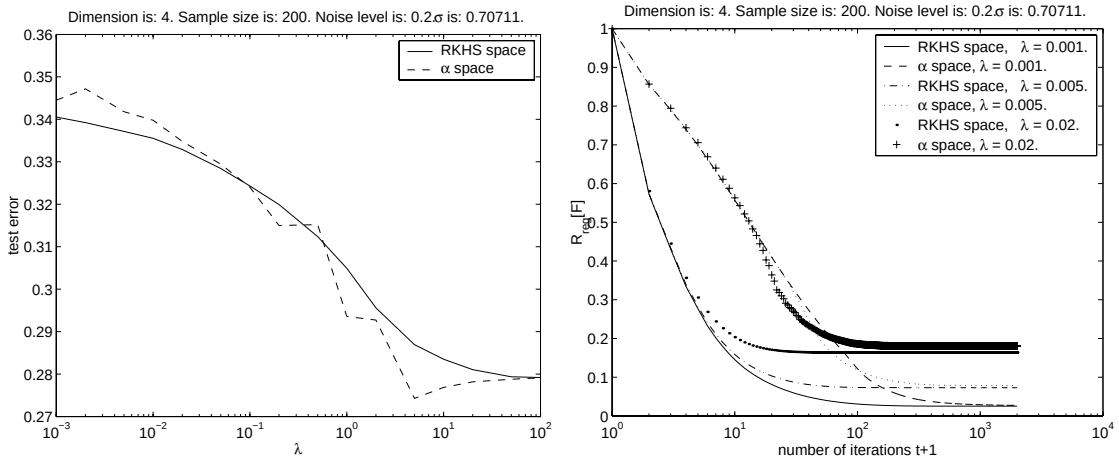

Figure 6: Left: test error versus λ. Solid curve: Gradient in Function Space. Dashed curve: Gradient in Coefficient Space. Right: R_{reg} versus the iteration time $t + 1$ for RKHS NormBoost Algorithm.

The experiments also show that the dimension and the sample size influence the convergence performance of $R_{reg}[f]$. Figure 8 shows that for low sample size, the convergence speed of $R_{reg}[f]$ shows a big difference between the gradient in function space and in coefficient space for low dimensions, but it is hard to tell the difference for high dimension

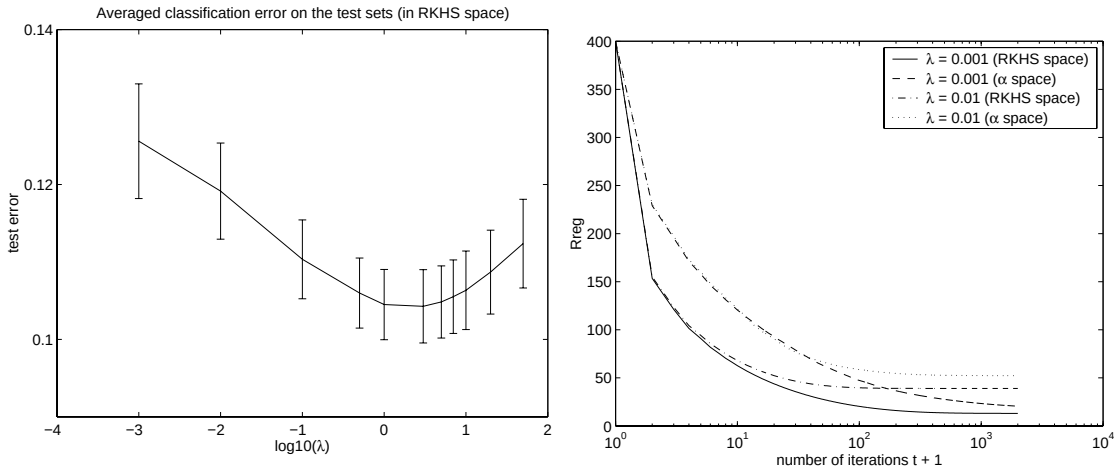

Figure 7: Left: test error versus λ. Right: R_{reg} versus the iteration time $t+1$ for RKHS NormBoost algorithm for BANANA dataset. Solid curve: Gradient in function space. Dashed curve: Gradient in coefficient space.

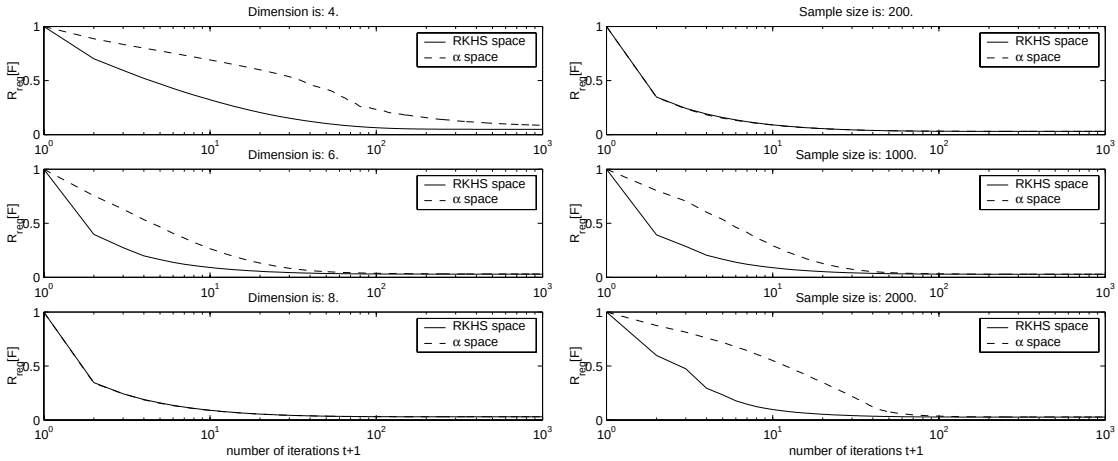

Figure 8: R_{reg} versus the iteration time t. Left: change the function dimension and keep the sample size fixed (sample size = 200). Right: change the sample size and keep the function dimension fixed (dimension = 8).

(dimension=8). On the other hand, if we increase the sample size, then the difference appears for high dimensions (see the right figure of Figure 8).

5. Discussion

Both the experiments in $\ell_2^{\mathbf{X}^m}$ NormBoost and RKHS NormBoost show a significant improvement in the convergence speed, which is one benefit of the regularization term. We

described the regularization term as a smoothness function in Section 2, where one function is said to be smoother than another one within a class of functions if it oscillates less. If we look at the functions in the frequency domain, a function is said to be smoother than another one if it has less energy at high frequency. The high frequency content of a function can be measured by first high-pass filtering the function, and then measuring the power. This means that we compute the norm of the function after the high-pass filtering (Girosi et al., 1995). Hence, the minimizer of the regularized risk functional (14) has not only less cost on the training samples, but also less high frequency content. In the frequency domain, such an optimal point in function space is easier to find than another one with more high frequency content. Hence adding a regularization term leads to much faster convergence than the standard AdaBoost algorithm.

For the RKHS NormBoost, there is another important reason for such remarkably faster convergence speed. Based on Theorem 1, the optimal regularized risk functional contains no more than m terms, where m is the sample size. This is normally much less than the number of iterations T in standard boosting algorithms. Hence the search for the optimal combination need only be performed in an m-dimensional space instead of dealing with a possibly infinite dimensional space (see Figure 5).

A curious feature of the graphs in Figure 1 for non-zero noise levels is that there is a discontinuity at $\lambda = 0$. Although as we have argued we need to regularize when there is noise, and thus want to set $\lambda > 0$, it is an effect that cries out for an explanation. We conjecture that with no regularization the loss surface is rank degenerate and the gradient descent procedure implemented by the NormBoost algorithm essentially drifts around a subspace in directions where the gradient is zero. As soon as one sets $\lambda > 0$, this degeneracy disappears. In numerical experiments we have found that setting λ to even extremely small values (such as 10^{-12}) suffices to bring the test error away from the point marked by a $*$ in Figure 1.

6. Generalization Performance

Next we will analyze the generalization performance of two function classes in the $\ell_2^{\mathbf{X}^m}$ space. Before we go into further detail, let us define some notation. For $0 < p \leq \infty$ and $d \in \mathbb{N}$, we define the spaces $\ell_p^d := \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_{\ell_p^d} < \infty\}$, where the p-norms are

$$\|\mathbf{x}\|_{\ell_p^d}^p := \sum_{j=1}^{d} |x_j|^p, \text{ for } 0 < p < \infty; \text{ and } \|\mathbf{x}\|_{\ell_\infty^d} := \max_{j=1,\ldots,d} |x_j|, \text{ for } p = \infty.$$

For $d = \infty$, we write $\ell_p = \ell_p^\infty$ and the norms are defined similarly. We use $\mathcal{F}_1(\mathcal{H}, C)$ and $\mathcal{F}_2(\mathcal{H}, V)$ to denote two different function classes:

$$\mathcal{F}_1(\mathcal{H}, C) = \left\{ f := \sum_{t=1}^{T} \alpha_t f_t,\ T \in \mathbb{N},\ f_t \in \mathcal{H},\ \|\boldsymbol{\alpha}\|_{\ell_1^T} \leq C \right\},$$

$$\mathcal{F}_2(\mathcal{H}, V) = \left\{ f := \sum_{t=1}^{T} \alpha_t f_t,\ T \in \mathbb{N},\ f_t \in \mathcal{H},\ \|f\|_{\ell_2^{\mathbf{X}^m}} \leq V \right\},$$

where $\mathcal{H}$ is the base classifier space. Suppose $\|f\|_{\ell_\infty^{\mathbf{X}^m}} \leq B$ for all $f \in \mathcal{H}$, $\alpha_i \geq 0$ for all $i \in \mathbb{R}$. The following theorem shows that function classes $\mathcal{F}_1(\mathcal{H}, C)$ and $\mathcal{F}_2(\mathcal{H}, V)$ can be bounded by each other if we add a condition on the inner product $\langle f_i, f_j \rangle_{\ell_2^{\mathbf{X}^m}}$.

Theorem 3 *Given m points $\mathbf{x}_1, \ldots, \mathbf{x}_m \in X$, $\mathcal{F}_1(\mathcal{H}, C)$ and $\mathcal{F}_2(\mathcal{H}, V)$ are defined as above. Suppose $\langle f_i, f_j \rangle_{\ell_2^{\mathbf{X}^m}} \geq \beta^2$ for all distinct f_i, f_j in $\mathcal{H}$, where $\beta > 0$. Then the relationship between these two function classes is:*

$$\mathcal{F}_1(\mathcal{H}, C) \subseteq \mathcal{F}_2\left(\mathcal{H}, BC\right), \tag{46}$$

and

$$\mathcal{F}_2(\mathcal{H}, V) \subseteq \mathcal{F}_1\left(\mathcal{H}, \frac{V}{\beta}\right). \tag{47}$$

Proof Part A: to prove the inequality (46). From the definition of the inner product, we have

$$\begin{aligned}
\|f\|_{\ell_2^{\mathbf{X}^m}}^2 = \langle f, f \rangle &= \frac{1}{m} \sum_{i=1}^{m} f^2(\mathbf{x}_i) \\
&= \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{j=1}^{T} \alpha_j f_j(\mathbf{x}_i) \sum_{k=1}^{T} \alpha_k f_k(\mathbf{x}_i) \right) \\
&= \sum_{j=1}^{T} \sum_{k=1}^{T} \alpha_j \alpha_k \left(\frac{1}{m} \sum_{i=1}^{m} f_j(\mathbf{x}_i) f_k(\mathbf{x}_i) \right) \\
&= \sum_{j=1}^{T} \sum_{k=1}^{T} \alpha_j \alpha_k \langle f_j, f_k \rangle. \qquad (48)
\end{aligned}$$

Since $\|h\|_{\ell_\infty^{\mathbf{X}^m}} \leq B$ for all $h \in \mathcal{H}$, we can get:

$$\langle f_j, f_k \rangle_{\ell_2^{\mathbf{X}^m}} = \frac{1}{m} \sum_{i=1}^{m} f_j(\mathbf{x}_i) f_k(\mathbf{x}_i) \leq \frac{1}{m} \left(m B^2 \right) = B^2.$$

Also noticing that $\|\alpha\|_{\ell_1^T} \leq C$ holds for the $f \in \mathcal{F}_1(\mathcal{H}, C)$, (48) can be written as

$$\|f\|_{\ell_2^{\mathbf{X}^m}}^2 \leq B^2 \left(\sum_{j=1}^{T} \sum_{k=1}^{T} \alpha_j \alpha_k \right) = B^2 \left(\sum_{j=1}^{T} \alpha_j \right)^2 \leq B^2 C^2$$

Hence, (46) is proved.

Part B: to prove the inequality (47). For the function $f \in \mathcal{F}_2(\mathcal{H}, V)$, we have $\|f\|_{\ell_2^{\mathbf{X}^m}} \leq V$. Combining with equation (48), we get

$$V^2 \geq \|f\|_{\ell_2^{\mathbf{X}^m}}^2 = \sum_{j=1}^{T} \sum_{k=1}^{T} \alpha_j \alpha_k \langle f_j, f_k \rangle.$$

Using the condition $\langle f_i, f_j \rangle_{\mathbf{X}^m} \geq \beta^2$, we have

$$V^2 \geq \beta^2 \sum_{j=1}^{T} \sum_{k=1}^{T} \alpha_j \alpha_k = \beta^2 \left(\sum_{j=1}^{T} \alpha_j \right)^2 = \beta^2 \|\alpha\|_{\ell_1^T}^2.$$

Hence, $\|\alpha\|_{\ell_1^T} \leq \frac{V}{\beta}$ holds, which implies (47). ■

From Theorem 3, we can see that if $\beta = 1$, i.e. $\forall i, j$, $\langle f_i, f_j \rangle_{\ell_2^{\mathbf{X}^m}} = 1$, $\mathcal{F}_1 = \mathcal{F}_2$ holds. In fact, the condition $\beta > 0$ is a very strong assumption which means that all the base function f_i are similar on the sample.

If we apply Theorem 3 with the theorem in Schapire et al. (1998), we obtain the following upper bound on the generalization error.

Corollary 4 *Let* $\mathbf{P}$ *be a distribution over* $X \times \{-1, 1\}$*, and let* S^m *be a sample of* m *examples chosen independently at random according to* $\mathbf{P}$*. Let* $\mathbf{P}_S\{\cdot\}$ *denote probability with respect to choosing an example uniformly at random from the training set* S^m*. Suppose the base-classifier space* $\mathcal{H}$ *has VC-dimension* d*, and let* $\delta > 0$*. Assume that* $m \geq d \geq 1$*,* $f \in \mathcal{F}_2$ *where* $\mathcal{F}_2$ *is defined as in Theorem 3 and* $\langle f_i, f_j \rangle_{\ell_2^{\mathbf{X}^m}} \geq \beta^2$ *where* $\beta > 0$*. Then with probability at least* $1 - \delta$ *over the random choice of the training set* S^m*, every weighted average function* f*, satisfies the following bound for all* $\theta > 0$*:*

$$\mathbf{P}\{(\mathbf{x}, y) : yf(\mathbf{x}) \leq 0\} \leq \mathbf{P}_S\{yf(\mathbf{x}) \leq \theta\} + O\left(\sqrt{\frac{dV^2 \log^2(m/d)}{m\beta^2\theta^2} + \frac{\log(1/\delta)}{m}} \right). \qquad (49)$$

Acknowledgments

The authors thank Jonathan Baxter and Bernhard Schölkopf for useful discussions. This research was supported by the Australian Research Council. A.S. was also supported by a grant of the DFG Sm 62/1-1. This research was carried out while Peter L. Bartlett was at the Australian National University.

References

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

Y. Bengio, Y. LeCun, and D. Henderson. Globally trained handwritten word recognizer using spatial representation, convolutional neural networks and hidden markov models. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 937–944, 1994.

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

L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1518, 1999.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Annals of Statistics*, 18:1676–1695, 1990.

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

N. Duffy and D. Helmbold. A geometric approach to leveraging weak learners. In *Computational Learning Theory: 4th European Conference*, volume 1572 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 1999.

M. Frean and T. Downs. A simple cost function for boosting. Technical report, Dept. of Computer Science and Electrical Engineering, University of Queensland, 1998.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Eurocolt '95*, pages 23–37. Springer-Verlag, 1995.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):400–407, 2000.

F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

P. J. Huber. *Robust Statistics*. John Wiley and Sons, New York, 1981.

Michael Kearns and Leslie Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.

G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematics Analysis Application*, 33:82–95, 1971.

L. Mason, J. Baxter, P.L. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, Cambridge, MA, 2000. MIT Press. 221 – 246.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42 (3):287–320, 2001.

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

R. Schapire, Y. Freund, P. L. Bartlett, and W. Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.

B. Schölkopf, R. Herbrich, A. J. Smola, and R. C. Williamson. A generalized representer theorem. Technical Report 2000-81, NeuroCOLT, 2000. URL `http://www.neurocolt.com/abs/2000/abs00081.html`. To appear in *Proceedings COLT'2001*.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2001. Forthcoming.

A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics. Doklady*, 4:1035–1038, 1963.

A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill–Posed Problems*. Winston, Washington, DC, 1977.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, N.Y., 1995. ISBN 0-387-94559-8.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

G. Wahba. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. Technical Report 984, Department of Statistics, University of Wisconsin, Madison, 1997.

C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*. Kluwer, 1998.

C.K.I. Williams and D. Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(12):1342 – 1351, 1998.