Sparse Regression Ensembles in Infinite and Finite Hypothesis Spaces

Gunnar Rätsch (raetsch@first.gmd.de)
GMD FIRST, Kekulèstr. 7, 12489 Berlin, Germany

Ayhan Demiriz (demira@rpi.edu)
Dept. of Decision Sciences and Eng. Systems, Rensselaer Polytechnic Institute, Troy, NY 12180 USA

Kristin P. Bennett (bennek@rpi.edu)
Dept. of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180 USA

Abstract. We examine methods for constructing regression ensembles based on a linear program (LP). The ensemble regression function consists of linear combinations of base hypotheses generated by some boosting-type base learning algorithm. Unlike the classification case, for regression the set of possible hypotheses producible by the base learning algorithm may be infinite. We explicitly tackle the issue of how to define and solve ensemble regression when the hypothesis space is infinite. Our approach is based on a semi-infinite linear program that has an infinite number of constraints and a finite number of variables. We show that the regression problem is well posed for infinite hypothesis spaces in both the primal and dual spaces. Most importantly, we prove there exists an optimal solution to the infinite hypothesis space problem consisting of a finite number of hypothesis. We propose two algorithms for solving the infinite and finite hypothesis problems. One uses a column generation simplex-type algorithm and the other adopts an exponential barrier approach. Furthermore, we give sufficient conditions for the base learning algorithm and the hypothesis set to be used for infinite regression ensembles. Computational results show that these methods are extremely promising.

Keywords: Ensemble Learning, Boosting, Regression, Sparseness, Semi-Infinite Programming

1. Introduction

The past years have seen strong interest in boosting and other ensemble learning algorithms due to their success in practical classification applications (e.g. Drucker et al., 1993; LeCun et al., 1995; Maclin & Opitz, 1997; Schwenk & Bengio, 1997; Bauer & Kohavi, 1999; Dietterich, 1999). The basic idea of boosting (and ensemble learning in general) is to iteratively generate a sequence $\{h_t\}_{t=1}^T$ of functions (hypotheses) that are usually combined as

$$f_{\boldsymbol{\alpha}}(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}),$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_T]$ are the hypothesis coefficients used. The hypotheses h_t are elements of a hypothesis class $\mathcal{H} = \{h_j : j \in \mathcal{P}\}$, where $\mathcal{P}$ is the index set of hypotheses producible by a base learning algorithm L. Typically one assumes that the set of hypotheses $\mathcal{H}$ is finite, but we will also consider extensions to infinite hypothesis sets. For classification, the ensemble generates the label by $\mathrm{sign}(f_{\boldsymbol{\alpha}}(\mathbf{x}))$, which is the weighted majority of the votes. For regression, the predicted value is $f_{\boldsymbol{\alpha}}(\mathbf{x})$.

Recent research in this field has focused on the better understanding of these methods and on extensions that are concerned with robustness issues (Mason et al., 1998; Bennett et al., 2000; Rätsch et al., 2000b, 2001). It has been shown that most classification ensemble methods can be viewed as minimizing some function of the classification margin. Typically this is performed algorithmically using a gradient descent approach in function space. Recently, it has been shown that the soft margin maximization techniques utilized in support vector machines can be readily adapted to produce ensembles for classification (Bennett et al., 2000; Rätsch et al., 2000b). These algorithms optimize the soft margin and error measures originally proposed for support vector machines. For certain choices of error and margin norms, the problem can be formulated as a linear program (LP). At first glance, the LP may seem intractable since the number of variables in the linear program is proportional to the size of the hypothesis space which can be exponentially large. But in fact, two practical algorithms exist for optimizing soft margin ensembles. The first uses column generation in a simplex algorithm (Bennett et al., 2000). The second uses barrier functions in an interior-point method (Rätsch et al., 2000). The advantage of these linear programming approaches is that they produce sparse ensembles using fast finite algorithms. The purpose of this work is to tackle regression ensembles using the analogous support vector linear programming methodology for regression.

To date, relatively few papers have addressed ensembles for regression (Friedman, 1999; Duffy & Helmbold, 2000; Bertoni et al., 1997; Zemel & Pitassi, 2001). One major difficulty is rigorously defining the regression problem in an infinite hypothesis space. For classification assuming each hypothesis has a finite set of possible outputs, the hypothesis space is always finite since there are only a finite number of ways to label any finite training set. For regression, even relatively simple hypothesis spaces, such as linear functions constructed using weighted least squares, consist of an uncountable infinite set of hypotheses. It is not a priori clear how to even express a regression problem in an infinite hypothesis space. Clearly we can only practically consider ensemble functions that are a linear combination of some finite subset of the set of possible hypotheses.

In this work, we study directly the issue of infinite hypothesis spaces. We begin in Section 2 with a review of boosting type algorithms for classification and regression and examine the relationship between ensemble methods and linear programming. In Section 3, we review a linear program approach to sparse regression and show how it is easily extendible to ensemble regression for the finite hypothesis case. In Section 3.2 we investigate the dual of this linear program for ensemble regression. In Section 3.3, we propose a semi-infinite linear program formulation for "boosting" of infinite hypothesis sets, first in the dual and then in the primal space. The dual problem is called semi-infinite because it has an infinite number of constraints and a finite number of variables. An important sparseness property of the semi-infinite regression problem is that it has a solution consisting of a finite number of hypotheses. In Section 4, we propose two different algorithms for efficiently computing optimal ensembles. The exact implementation of these algorithms is dependent on the choice of base learning algorithms. In Section 4.3 we investigate three possible base learning algorithms that result in both infinite and finite hypothesis sets. Computational results are presented in Section 5.

The notational conventions used in this paper can be found in Table I.

Table I. Notational conventions

n, N	counter and number of patterns
j, J	counter and number of hypotheses if finite
t, T	counter and number of iterations
$\mathbf{p}, \mathcal{P}$	index and index-set for hypotheses
$\mathcal{X}, s$	input space, dimensionality of $\mathcal{X}$
X, Y, Z	training data: input, targets, both
$\mathbf{x}, y$	a training pattern and the label
$\mathcal{H}, h_j$	set of base hypotheses and an element of $\mathcal{H}$
$\mathcal{F}, f_{\boldsymbol{\alpha}}$	set of linear combinations of $\mathcal{H}$ and element of $\mathcal{F}$
$\boldsymbol{\alpha}$	hypothesis weight vector
$\mathbf{d}$	weighting on the training set
$\mathbf{w}$	a weight vector for linear models
$\mathrm{I}(\cdot)$	the indicator function: $\mathrm{I}(true) = 1$ and $\mathrm{I}(false) = 0$
ε	the tube size
ν	the tube parameter (determines ε)
C	the regularization (complexity) parameter
ϵ	weighted classification error
$\|\cdot\|_p$	the ℓ_p-norm, $p = [1, \infty]$
$\langle\cdot,\cdot\rangle, k(\cdot,\cdot)$	scalar product and scalar product in feature space

2. Boosting-type Algorithms

We briefly review and discuss existing boosting-type algorithms. In Section 2.1 we start with the classification case and describe AdaBoost (Freund & Schapire, 1996) and, closely related, Arc-GV (Breiman, 1997). Then we discuss properties of the solutions generated by boosting and show connections to a linear program (LP) for maximizing the margins. In Section 2.2 we briefly review some recent regression approaches that are mainly motivated from a gradient-descent understanding of Boosting.

2.1. Classification Boosting and LP

For the classification case, it is generally assumed the hypotheses class is $\mathcal{H} = \{h_j : \mathbf{x} \mapsto \{\pm 1\},\ j = 1, \ldots, J\}$, defined by a base learning algorithm L. In each iteration the base learner is used to select the next hypothesis using certain criteria. The ensemble generates the label which is the weighted majority of the votes by $\mathrm{sign}(f_{\boldsymbol{\alpha}}(\mathbf{x}))$. Note that the hypothesis class is always finite because there at most 2^N distinct labelings of the training data.

Consider the AdaBoost algorithm. For more details see e.g. (Freund & Schapire, 1994; Breiman, 1997). The main idea of AdaBoost is to introduce weights d_n $(n = 1, \ldots, N)$ on the training patterns $Z := \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$. They are used to control the importance of each single pattern for learning a new hypothesis (i.e., while repeatedly running the base algorithm). Training patterns that are difficult to learn (which are misclassified repeatedly) become more important by increasing their weight.

It has been shown that AdaBoost minimizes an error function (Breiman, 1997; Frean & Downs, 1998; Friedman et al., 1998; Rätsch et al., 2001) that can be expressed in terms of margins, namely it iteratively solves the problem

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}} & G(\boldsymbol{\alpha}) := \sum_{n=1}^{N} \exp(-y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)) \\ \text{with} & \boldsymbol{\alpha} \geq 0 \end{array}$$

The optimization strategy of AdaBoost has also been called "gradient descent" in function space (Mason et al., 1999; Friedman et al., 1998), as one effectively optimizes along restricted gradient directions in the space of linearly combined functions f. This can also be understood as a coordinate descent method (e.g. Luenberger, 1984) to minimize $G(\boldsymbol{\alpha})$ over all possible weightings of hypotheses from $\mathcal{H}$ (Rätsch et al., 2000). One hypothesis is added at a time and its weight is never changed unless the same hypothesis is added again.

It is widely believed (Breiman, 1997; Freund & Schapire, 1996; Schapire et al., 1997; Rätsch et al., 2001, 2001) that AdaBoost approximately maximizes the *smallest margin*, ϱ

$$\varrho(\boldsymbol{\alpha}) := \min_{1 \leq n \leq N} \frac{y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)}{\|\boldsymbol{\alpha}\|_1} ,$$

on the training set. This problem can be solved exactly by the following linear programming problem over the complete hypothesis set $\mathcal{H}$ (cf. Grove and Schuurmans (1998), assuming a *finite* number of basis hypotheses):

$$\begin{array}{lrl} \max\limits_{\varrho,\boldsymbol{\alpha}} & \varrho & \\ \text{with} & y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n) \geq \varrho & \text{for all } 1 \leq n \leq N \\ & \alpha_j, \varrho \geq 0 & \text{for all } 1 \leq j \leq J \\ & \|\boldsymbol{\alpha}\|_1 = 1 & \end{array} \tag{1}$$

Breiman (1997) proposed a modification of AdaBoost – Arc-GV – making it possible to show the asymptotic convergence of $\varrho(\boldsymbol{\alpha}^t)$ $(t \to \infty)$ to a global solution ϱ^{lp} of (1). In Grove and Schuurmans (1998) the LP (1) was solved using an iterative linear programming based approach that retrospectively can be considered as a column generation algorithm. Unfortunately, neither approach performed well in practice.

Soft margin versions of this linear program based on ideas from support vector machines perform very well both in practice and theoretically in terms of generalization bounds (Rätsch et al., 2000b; Bennett et al., 2000). For example, a soft margin version could be

$$\begin{array}{lrl} \max\limits_{\varrho,\boldsymbol{\alpha},\boldsymbol{\xi}} & \varrho - C\sum_{n=1}^{N}\xi_n & \\ \text{with} & y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n) + \xi_n \geq \varrho & \text{for all } 1 \leq n \leq N \\ & \boldsymbol{\alpha}, \boldsymbol{\xi}, \varrho \geq 0 & \|\boldsymbol{\alpha}\|_1 = 1 \end{array}$$

In Bennett et al. (2000) the column generation algorithm for classification was proposed to efficiently solve these LPs. This algorithm and those closely related in Rätsch et al. (2001), Kivinen and Warmuth (1999) differ from the gradient-boosting idea used to motivate boosting-type algorithms (Mason et al., 1999; Friedman et al., 1998). At each iteration, all generated hypothesis weights are optimized with respect to a maximum margin error function. The gradient approach fixes hypothesis weights as the hypotheses are generated. The purpose of this paper is to examine the extensions of these approaches to the regression case.

2.2. Previous Regression Approaches

Several regression boosting methods have been proposed. We provide a brief description of three of them. Note that the first two described here and also those of (Fisher, 1997; G. Ridgeway, 1999) reduce the problem to a series of classification tasks, thus eliminating any consideration of infinite hypothesis spaces. The last approach (Friedman, 1999) has been applied to infinite hypothesis spaces, but does not define what it means to boost in an infinite hypothesis space.

2.2.1. *AdaBoost-R:*

The first boosting-type algorithm for regression – AdaBoost.R – was proposed in Freund and Schapire (1994). It is based on a reduction to the classification case. The algorithm aims to find a regression function $f : \mathbf{x} \mapsto [0, 1]$. A problem with this algorithm is that it uses a piece-wise linear function on $[0, 1]$ whose number of branch-points increases exponentially with the number of iterations. Therefore, the algorithm is computationally intractable.

2.2.2. *AdaBoost-RΔ:*

Another reduction for finding $f : \mathbf{x} \mapsto [0, 1]$ to the classification case was proposed in Bertoni et al. (1997). Here, a pattern that is predicted with error less than some $\Delta > 0$ is counted as correctly classified and as misclassified otherwise. The combined regression function is given by

$$f(\mathbf{x}) = \operatorname*{argmax}_{y \in [0,1]} \sum_{t=1}^{T} \alpha_t \mathrm{I}(|h_t(\mathbf{x}) - y| \leq \Delta).$$

Again, a probability weighting $\mathbf{d}$ on the training patterns is used. Under the assumption that the weighted "classification error" $\epsilon = \sum_{n=1}^{N} d_n \mathrm{I}(|h_t(\mathbf{x}_n) - y_n| \geq \Delta)$ in each iteration is smaller than $\frac{1}{2} - \gamma$ ($\gamma > 0$), the number of training patterns for which $|f(\mathbf{x}_n) - y_n| \geq 2\Delta$ converges quickly to zero. From our experience it turned out that (i) the choice of Δ is rather difficult and (ii) the selection of the next hypothesis by the base learner is a demanding problem, as the weighted error ϵ usually converges quickly to $\frac{1}{2}$ and the algorithm has to stop.

2.2.3. *Gradient Boosting for Regression (Friedman, 1999):*

Based on the understanding of boosting as a gradient descent method, other regression algorithms have been proposed – e.g. in the very interesting paper of Friedman (Friedman, 1999). Here, the derivative $\frac{\partial G}{\partial f(\mathbf{x}_n)}$ of a cost function G (e.g. squared loss: $G = \sum_{n=1}^{N}(f(\mathbf{x}_n) - y_n)^2$) is taken with respect to the output $f(\mathbf{x}_n)$ of the regression function. Then the projected gradient direction (a basis function $h \in \mathcal{H}$) that is most in the direction of the true gradient is found by

$$(h, \alpha) = \operatorname*{argmax}_{h\in\mathcal{H},\alpha\in\mathbb{R}} \sum_{n=1}^{N} \left(\frac{\partial G}{\partial f(\mathbf{x}_n)} - \alpha h(\mathbf{x}_n) \right)^2 . \qquad (2)$$

This idea has been worked out for squared loss, linear absolute loss, and Huber's loss. However, the gradient direction found in (2) is optimal for the squared loss only. For the linear absolute loss, this has been specialized to the Tree-Boost algorithm (Friedman, 1999). Here, the task of finding the next hypothesis is posed as a classification problem, where the sign of the gradient determines the class membership. In this algorithm, the aim is to maximize the correlation between the gradient and the output of the base hypothesis. This approach is similar to the algorithm proposed in Section 4.3.3.

This approach works well in practice. It does not explicitly deal with the infinite hypothesis case. Like all gradient descent algorithms it offers convergence only in the limit – even for finite hypothesis spaces. Since regularization is not used, it can potentially overfit so development of good stopping criteria is essential. In the next section, we will develop an alternative approach based on linear programming. The advantages of the LP approach include extensibility to the infinite hypothesis case, sparse solution, guarantee of the existence of sparse finite solutions, and practical fast finite algorithms.

3. Linear Programs for Regression

In this section, we develop finite and semi-infinite LP formulations for the sparse ensemble regression. We begin with the primal LP for the finite case, then investigate the dual finite LP. Then we extend this to the dual and primal infinite hypothesis cases.

3.1. Finite Sparse Linear Regression

Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in \mathcal{X} \times \mathbb{R}$ be some i.i.d. (training) data. The regression problem is often stated as finding a function $f^* \in \mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}\}$ that minimizes the regularized risk functional (Vapnik, 1995; Smola, 1998)

$$R[f] := \mathbf{P}[f] + \frac{C}{N} \sum_{n=1}^{N} l(y_n - f(\mathbf{x}_n)), \quad \text{with} \quad f^* = \operatorname*{argmin}_{f\in\mathcal{F}} R[f], \quad (3)$$

where $l(\cdot)$ is a loss function, $\mathbf{P}[\cdot]$ a regularization operator, and C the regularization parameter, determining the trade-off between loss and complexity (i.e., size of the function class).

In this paper we consider the well-known ε-insensitive loss (Vapnik, 1995; Schölkopf et al., 1999) as loss function:

$$l_\varepsilon(y - f(\mathbf{x})) := |y - f(\mathbf{x})|_\varepsilon = \max(0, |y - f(\mathbf{x})| - \varepsilon) \tag{4}$$

This does not penalize errors below some $\varepsilon \geq 0$, chosen a priori. It has been shown to have several nice properties – as we will see later (cf. Smola, 1998). However, in principle the analysis and algorithms also work for other loss functions (cf. Rätsch, 2001).

In this paper we consider $\mathcal{F}$ to be the space of linear combinations of base hypotheses of another space $\mathcal{H}$ – the so-called *base hypothesis space* – including a bias, i.e.

$$\mathcal{F} = \left\{ f_{\boldsymbol{\alpha}} \;\middle|\; f_{\boldsymbol{\alpha}}(\mathbf{x}) = b + \sum_{j=1}^{J} \alpha_j h_j(\mathbf{x}),\; b \in \mathbb{R}, \mathbf{0} \leq \boldsymbol{\alpha} \in \mathbb{R}^J \right\} \tag{5}$$

Here we assume $\mathcal{H}$ has a finite number of hypotheses (J of them). This will be generalized to infinite hypothesis classes in Sections 3.3 and 3.4. Throughout the paper we assume that $\mathcal{H}$ is closed under complementation ($h \in \mathcal{H} \;\Rightarrow\; -h \in \mathcal{H}$). Hence, one may enforce $\alpha_j \geq 0$ without effectively changing $\mathcal{F}$.

Let us consider the ℓ_1-norm of the hypothesis coefficients as a regularization operator, i.e. $\mathbf{P}_1[f_{\boldsymbol{\alpha}}] := \|\boldsymbol{\alpha}\|_1$. Using (4), minimizing (3) can be stated as a linear program, which we call the *LP-Regression problem*:

$$\begin{array}{lll}
\min\limits_{\boldsymbol{\alpha}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*} & \|\boldsymbol{\alpha}\|_1 + \dfrac{C}{N} \left(\sum\limits_{n=1}^{N} \xi_n + \xi_n^* \right) & \\
\text{with} & y_n - f_{\boldsymbol{\alpha}}(\mathbf{x}_n) \leq \varepsilon + \xi_n & n = 1, \ldots, N \\
 & f_{\boldsymbol{\alpha}}(\mathbf{x}_n) - y_n \leq \varepsilon + \xi_n^* & n = 1, \ldots, N \\
 & \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\xi}^* \geq \mathbf{0}, & \\
 & \boldsymbol{\alpha} \in \mathbb{R}^J, b \in \mathbb{R}, \boldsymbol{\xi}, \boldsymbol{\xi}^* \in \mathbb{R}^N, &
\end{array} \tag{6}$$

where $f_{\boldsymbol{\alpha}}(\mathbf{x}) = b + \sum_{j=1}^{J} \alpha_j h_j(\mathbf{x})$ as in (5) and $\varepsilon \geq 0$ is a fixed constant. The regularization operator $||\boldsymbol{\alpha}||_1$ is frequently used in sparse favoring approaches, e.g. basis pursuit (Chen et al., 1995) and parsimonious least norm approximation (Bradley et al., 1998). Roughly speaking, a reason for the induced sparseness is the fact that vectors far from the coordinate axes are "larger" with respect to the ℓ_1-norm than with respect to 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. It can easily be shown (cf. Corollary 4) that independent of the size of a (finite) hypothesis space $\mathcal{H}$, the optimal number of hypotheses in the ensemble is not greater than the number of samples. The optimization algorithms proposed in Section 4 exploit this property.

A nice property of (6) is that its solution is robust with respect to small changes of the training data:

Proposition 1 (Smola et al. (1999)). *Using Linear Programming Regression with the ε-insensitive loss function (4), local movements of target values of points inside and outside (i.e. not on the edge of) the ε-tube do not influence the regression.*

The parameter ε in (6) is usually difficult to control (Müller et al., 1997; Schölkopf et al., 2000), as one usually does not know beforehand how accurately one is able to fit the curve. This problem is partially resolved in the following optimization problem (Smola et al., 1999) for $\nu \in (0, 1]$:

$$\begin{array}{lll} \min\limits_{\boldsymbol{\alpha}, b, \varepsilon, \boldsymbol{\xi}, \boldsymbol{\xi}^*} & \|\boldsymbol{\alpha}\|_1 + C\dfrac{1}{N}\left(\sum\limits_{n=1}^{N} \xi_n + \xi_n^*\right) + C\nu\varepsilon & \\ \text{with} & y_n - f_{\boldsymbol{\alpha}}(\mathbf{x}_n) \le \varepsilon + \xi_n & n = 1, \ldots, N \\ & f_{\boldsymbol{\alpha}}(\mathbf{x}_n) - y_n \le \varepsilon + \xi_n^* & n = 1, \ldots, N \\ & \boldsymbol{\alpha} \ge \mathbf{0}, \varepsilon \ge 0, \boldsymbol{\xi}, \boldsymbol{\xi}^* \ge \mathbf{0} & \\ & b \in \mathbb{R}, \boldsymbol{\xi}, \boldsymbol{\xi}^* \in \mathbb{R}^N, \boldsymbol{\alpha} \in \mathbb{R}^J. & \end{array} \tag{7}$$

The difference between (6) and (7) lies in the fact that ε has become a positively constrained variable of the optimization problem itself. The core aspect of (7) can be captured in the proposition stated below.

Proposition 2 (Smola et al. (1999)). *Assume $\varepsilon > 0$. The following statements hold:*

(i) ν is an upper bound on the fraction of errors (i.e. points outside the ε tube).

(ii) ν is a lower bound on the fraction of points not inside (i.e. outside or on the edge of) the ε tube.

(iii) Suppose the data were generated i.i.d. from a distribution $P(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$ with $P(y|\mathbf{x})$ continuous. With probability 1, asymptotically, ν equals both the fraction of points not inside the tube and the fraction of errors.

Summarizing, the optimization problem (7) has two parameters: (i) the regularization parameter C, which controls the size of the hypothesis set and therefore the complexity of the regression function, and (ii) the *tube-parameter* ν, which directly controls the fraction of patterns outside the ε-tube and indirectly controls the size of the ε-tube.

3.2. Dual Finite LP Formulation

In this section we state the dual optimization problem of (7) by introducing Lagrangian multipliers d_n for the first constraint which computes the error if the target is underestimated, and d_n^* which the error measures if the target is overestimated. See any linear programming text book for specifics on how to construct a dual LP problem.

The dual problem of (7) is

$$\begin{array}{lll} \max\limits_{\mathbf{d},\mathbf{d}^*} & \sum\limits_{n=1}^{N} y_n(d_n - d_n^*) & \\ \text{with} & \sum_n d_n - d_n^* = 0 & \\ & \sum_n d_n + d_n^* \leq C\nu & \\ & \sum_n h_j(\mathbf{x}_n)(d_n - d_n^*) \leq 1 & j = 1, \ldots, J \\ & 0 \leq d_n, d_n^* \leq C/N & n = 1, \ldots, N, \end{array} \tag{8}$$

where the constraint $\sum_n d_n + d_n^* \leq C\nu$ comes from the reparameterization of ε with ν. Here, we have $2N + 2$ fixed constraints and $J := |\mathcal{H}|$ constraints, one for each hypothesis $h \in \mathcal{H}$. At optimality for each point, the quantity $p_n = d_n - d_n^*$ defines an error residual. By complementarity, we know that if the ε-error is zero (that is if $-\varepsilon < f_{\boldsymbol{\alpha}}(\mathbf{x}_n) - y_n < \varepsilon$), then $d_n = d_n^* = 0$. If the point is underestimated, $-\varepsilon > f_{\boldsymbol{\alpha}}(\mathbf{x}_n) - y_n$, then $d_n \geq 0$ and $d_n^* = 0$. Likewise, if the point is overestimated, $f_{\boldsymbol{\alpha}}(\mathbf{x}_n) - y_n > \varepsilon$, then $d_n = 0$ and $d_n^* \geq 0$. Thus $p_n = 0$ if the point is within the ϵ-tube, $p_n > 0$ when the point falls below the ϵ-tube, and $p_n < 0$ if the point falls above the ϵ-tube. The magnitude of p_n reflects the sensitivity of the objective to changes in ϵ. The larger the change in error, the larger p_n. The quantity in the constraints $\sum_n h(\mathbf{x}_n)(d_n - d_n^*)$ reflects how well the hypothesis addressed the residual errors. If $\sum_n h(\mathbf{x}_n)(d_n - d_n^*)$ is positive and large in size then the hypothesis will be likely to improve the ensemble. But it must be sufficiently large to offset the penalty for increasing $\|\boldsymbol{\alpha}\|_1$.

3.3. Generalization to Infinite Hypotheses

Consider now the case where there is an infinite set of possible hypotheses $\mathcal{H}$. Say we select any finite subset H_1 of $\mathcal{H}$, then the primal and dual regression LPs on H_1 are well defined. Now say we increase the subset size and define $H_2 \supset H_1$ of $\mathcal{H}$. What is the relationship between the optimal ensembles created on the two subsets? A solution of the smaller H_1 LP is always primal feasible for the larger H_2 LP. If the H_1 solution is dual feasible for the larger H_2 LP, then the solution is also optimal for the problem H_2. So dual feasibility is the key issue. Define the base learning algorithm L for a fixed $\mathbf{p}$ as

$$h_{\mathbf{p}} := L(X, \mathbf{p}) := \operatorname*{argmax}_{h \in \mathcal{H}} \sum_n h(\mathbf{x}_n) p_n \tag{9}$$

If $\sum_n h_{\mathbf{p}}(\mathbf{x}_n) p_n = \sum_n h_{\mathbf{p}}(\mathbf{x}_n)(d_n - d_n^*) > 1$, then dual feasibility is violated; $h_{\mathbf{p}}$ is a good hypothesis that should be added to the ensemble, and the solution may not be optimal.

By thinking of h as a function of $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and $\mathbf{p}$ as in (9), we can extend the dual problem (8) to the infinite hypotheses case. The set of dual feasible values of $\mathbf{p}$ is equivalent to the following compact polyhedron:

$$\mathcal{P} = \left\{ \mathbf{p} \;\middle|\; \sum_{n=1}^{N} |p_n| \leq C\nu, \sum_{n=1}^{N} p_n = 0, \ |p_n| \leq \frac{C}{N}, n = 1, \ldots, N \right\}. \tag{10}$$

The *dual SILP-regression problem* is

$$\begin{array}{rl} \Omega(D) = \max\limits_{\mathbf{d}, \mathbf{d}^*} & \sum\limits_{n=1}^{N} y_n (d_n - d_n^*) \\ \text{with} & \sum_{n=1}^{N} h_{\mathbf{p}}(\mathbf{x}_n)(d_n - d_n^*) \leq 1, \quad \forall \mathbf{p} \in \mathcal{P} \\ & \sum_{n=1}^{N} (d_n + d_n^*) \leq C\nu \\ & \sum_{n=1}^{N} (d_n - d_n^*) = 0 \\ & 0 \leq d_n, d_n^* \leq \frac{C}{N}, \ n = 1, \ldots, N \end{array} \tag{11}$$

This is an example of semi-infinite linear program (SILP), a class of problems that has been extensively studied in mathematical programming. The problem is called semi-infinite because it has an infinite number of constraints and a finite number of variables. The set $\mathcal{P}$ is known as the index set. If the set of hypotheses producible by the base learner is finite, e.g. if $\{h \mid h = L(X, \mathbf{p}), \ \mathbf{p} \in \mathcal{P}\}$ is finite, then the problem is exactly equivalent to LP-Regression problem (8).

We will establish several facts about this semi-infinite programming problem using the results for general linear semi-infinite programs summarized in the excellent review paper (Hettich & Kortanek, 1993). To simplify the presentation, we simplified the results in Hettich and Kortanek (1993) to the case of SILP with an additional set of finite linear constraints. The results presented can be easily derived from (Hettich & Kortanek, 1993) through a change in notation and by increasing the index set to include the additional finite set of traditional linear constraints. To be consistent with our derivation of the SILP-regression problem, we will refer to the problem with infinitely many constraints as the dual problem and the problem with infinitely many variables as the primal problem. Care should be taken, since this is the reverse of the convention used in the mathematical programming literature.

We define the generic dual SILP as

$$\Phi(D) = \max_{z \in \mathbb{R}^N} \{ \langle c, z \rangle \mid \langle a(\mathbf{p}), z \rangle \leq b(z), Qz \leq q, \ \forall \mathbf{p} \in B \} \tag{12}$$

where $c \in \mathbb{R}^N$, $Q \in \mathbb{R}^{r \times m}$, $q \in \mathbb{R}^r$, B and $\{z \mid Qz \leq q\}$ are compact sets, $a(\cdot)$ is a function from B to $\mathbb{R}^N$, and $b(\cdot)$ is a function from $\mathbb{R}^N$ to $\mathbb{R}$. We will make the additional assumption that the problem is always feasible and that the feasible region is compact. Clearly the maximum value is always obtained since we are maximizing a continuous function over a compact set.

Ideally, we would like the solution of a linear program to correspond to the optimal solution of the semi-infinite problem. We now define a necessary condition for the existence of a finite linear program whose optimal solution also solves the semi-infinite program. We will denote the generic dual SILP restricted to a finite subset $\mathcal{P}_N = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \subseteq B$ as $\Phi(D(\mathcal{P}_N))$. This is a linear program since it has a finite number of constraints.

The first theorem gives necessary conditions for the optimal solution of a generic dual SILP to be equivalent to the solution of a finite linear program (Theorem 4.2 in Hettich & Kortanek, 1993):

Theorem 3 (Necessary condition for finite solution). *Assume the following Slater condition holds: For every set of $N+1$ points, $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_N$, there exists $\hat{z}$ such that $\langle a(\mathbf{p}_n), \hat{z} \rangle < b(\mathbf{p}_n)$, $n = 1, \ldots, N$, and $Q\hat{z} < r$. Then there exists $\mathcal{P}_N = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \subseteq B$ such that*

1. *$\Phi(D) = \Phi(D(\mathcal{P}_N))$;*
2. *There exist multipliers $\mu_n \geq 0$, $n = 1, \ldots, N$, such that*

$$\Phi(D) = \min_{z \in \mathbb{R}^N} \left\{ \langle c, z \rangle - \sum_{n=1}^{N} \mu_n (a(\mathbf{p}_n) - b(\mathbf{p}_n)) \mid Qz \leq q \right\}. \tag{13}$$

This result immediately applies to the dual SILP regression problem since the strictly interior point $\mathbf{p} = \mathbf{d} - \mathbf{d}^*$ with $\mathbf{d}_n = \mathbf{d}_n^* = C\nu/(2N^2 + 1)$ satisfies the Slater condition.

Corollary 4 (Finite solution of regression ensemble). *For Problem $\Omega(D)$ (11) with $\nu < 1$, there exists $\mathcal{P}_N = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \in \mathcal{P}$ such that $\Omega(D) = \Omega(D(\mathcal{P}_N))$.*

The significance of this result is that there exists an optimial ensemble that consists of at most N hypotheses where N is the number of data points and that this is true even if the set of possible hypotheses is infinite.

3.4. Primal Regression SILP

Next we look at the corresponding primal problem for the semi-infinite case. We would like our semi-infinite dual problem to be equivalent to a meaningful primal problem that simplifies to the original primal for the finite hypothesis case.

Let $M^+(B)$ be the set of nonnegative Borel measures on B. The subset

$$\mathbb{R}_+^{(B)} := \{\mu \in M^+(B) \mid \mathrm{supp}(\mu) \text{ finite}\} \tag{14}$$

denotes the set of nonnegative generalized finite sequences. The primal problem of the generic SILP (12) is

$$\Phi(P) = \inf_{\mu \in \mathbb{R}_+^{(B)}} \left\{ \sum_{\mathbf{p} \in B} b(\mathbf{p})\mu(\mathbf{p}) \mid \sum_{\mathbf{p} \in B} a(\mathbf{p})\mu(\mathbf{p}) = c \right\} \tag{15}$$

In finite linear programming, the optimal objective values of the primal and dual problems are always equal. This is not always true for the semi-infinite case. Weak duality always holds, that is, $\Phi(P) \leq \Phi(D)$. We must ensure that there is no duality gap, i.e., that $\Phi(P) = \Phi(D)$. From Hettich and Kortanek (1993) (Theorem 6.5) we have the following

Theorem 5 (Sufficient conditions for no duality gap). *Let the convex cone*

$$\begin{aligned} M_{N+1} &= co\left(\left\{ \begin{pmatrix} a(\mathbf{p}) \\ b(\mathbf{p}) \end{pmatrix} \mid \mathbf{p} \in B \right\}\right) \in \mathbb{R}^{N+1} \\ &= \left\{ w = \textstyle\sum_{\mathbf{p} \in B} \lambda(\mathbf{p}) \begin{pmatrix} a(\mathbf{p}) \\ b(\mathbf{p}) \end{pmatrix},\ \lambda \in \mathbb{R}_+^{(B)} \right\} \in \mathbb{R}^{N+1}. \end{aligned} \tag{16}$$

be closed, then $\Phi(P) = \Phi(D)$ and primal minimum is attained.

For the regression problem, $a(\mathbf{p}) = h_{\mathbf{p}}(X) = [h_{\mathbf{p}}(\mathbf{x}_1), \ldots, h_{\mathbf{p}}(\mathbf{x}_N)]^\top$ is the set of base hypotheses (evaluated at the training points) obtainable by our learning algorithm, and $b(\mathbf{p}) = 1$ is constant. Thus the theorem can be simplified as follows.

Corollary 6 (Sufficient conditions for base learner). *Let the convex cone*

$$\begin{aligned} M_N &= co\left(\{h_{\mathbf{p}}(X) \mid \mathbf{p} \in \mathcal{P}\}\right) \in \mathbb{R}^N \\ &= \left\{ w = \textstyle\sum_{\mathbf{p} \in \mathcal{P}} \lambda(\mathbf{p}) h_{\mathbf{p}}(X),\ \lambda \in \mathbb{R}_+^{(\mathcal{P})} \right\} \in \mathbb{R}^N. \end{aligned} \tag{17}$$

be closed, then $\Omega(P) = \Omega(D)$ *and primal minimum is attained.*

This corollary imposes conditions on the set of possible base hypotheses. Some examples of sets of base hypothesis that would satisfy this condition are:

- The set of possible hypotheses is finite, e.g. $\{h \mid h = L(X, \mathbf{p}),\ \mathbf{p} \in \mathcal{P}\}$ is finite.

- The function $h_{\mathbf{p}} = L(X, \mathbf{p})$ is continuous with respect to $\mathbf{p}$.

These two conditions are sufficient to cover all the base hypotheses considered in this paper, but other conditions are possible.

4. LP Ensemble Optimization Algorithms

In this section we propose two algorithms for optimizing finite and infinite regression linear programs. The first uses column generation to execute a simplex-type algorithm. The second adopts an exponential barrier strategy that has connections to boosting algorithms for classification (Rätsch et al., 2000).

4.1. Column Generation Approach

The basic idea of Column Generation (CG) is to construct the optimal ensemble for a restricted subset of the hypothesis space. LP (8) is solved for a finite subset of hypotheses. It is called the *restricted master problem*. Then the base learner is called to generate a hypothesis $h_t = L(X, \mathbf{p})$ where $\mathbf{p} = \mathbf{d} - \mathbf{d}^*$. Assuming the base learner finds the best hypothesis satisfying condition (9), if $\sum_n h_t(\mathbf{x}_n)(\mathbf{d}_n - \mathbf{d}_n^*) < 1$ then the current ensemble is optimal as all constraints are fulfilled. If not, the hypothesis is added to the problem. This corresponds to generating a column in the primal LP or SILP or a row of the dual LP or SILP. The CG-Regression algorithm (cf. Algorithm 1) assumes that the base learner $L(X, \mathbf{p})$ is finite for any $\mathbf{p} \in \mathcal{P}$.

Algorithm 1 is a special case of the set of SILP algorithms known as **exchange methods**. These methods are known to converge. Clearly if the set of hypotheses is finite, then the method will converge in a finite number of iterations since no constraints are ever dropped. But one can also prove that it converges for SILP (cf. Theorem 7.2 in Hettich & Kortanek, 1993):

Theorem 7 (Convergence of Algorithm 1). *Algorithm 1 stops after a finite number of steps with a solution to the dual regression*

Algorithm 1 The CG-Regression algorithm.

argument: `Sample` $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \mathbf{y} = \{y_1, \ldots, y_N\}$
`Regularization constant` C`, Tube parameter` $\nu \in (0, 1)$
returns: `Linear combination from` $\mathcal{H}$.
function `CG-Reg`$(X, \mathbf{y}, C, \nu)$
  $t = 0$;
  repeat
    $t = t + 1$
    `Let` $[\mathbf{d}, \mathbf{d}^*]$ `be the solution of (8) using` $t - 1$ `hypotheses`
    $h_t := L(X, \mathbf{p})$`, where` $\mathbf{p} := \mathbf{d} - \mathbf{d}^*$
  until $\sum_n h_t(\mathbf{x}_n)(\mathbf{d}_n - \mathbf{d}_n^*) < 1$
  `Let` $[\boldsymbol{\alpha}, b]$ `be the dual solution to` $[\mathbf{d}, \mathbf{d}^*]$`, i.e. a solution to (7)`
  return $f = b + \sum_{q=1}^{t-1} \alpha_q h_q$
end

SILP or the sequence of intermediate solutions $(\mathbf{d}, \mathbf{d}^*)$ *has at least one accumulation point and each of these solves the dual regression SILP.*

This theorem holds for a more general set of exchange methods than Algorithm 1. For example, it is possible to add or drop multiple constraints at each iteration, and the convergence result is unchanged. In practice, we found the column generation algorithm stops at an optimal solution in a small number of iterations for both LP and SILP regression problems.

4.2. A Barrier Algorithm

In the following we propose an algorithm (see also Rätsch et al., 2000) that uses the barrier optimization technique (Bertsekas, 1995; Frisch, 1955; Cominetti & Dussault, 1994). For details on the connection between Boosting-type algorithms and barrier methods see Rätsch et al. (2000, 2001). A similar algorithm has been proposed in Duffy and Helmbold (2000), which has been developed independently. In this sequel, we will give a very brief introduction to barrier optimization.

The goal of barrier optimization is to find an optimal solution of the problem $\min_{\boldsymbol{\theta} \in \mathcal{S}} f(\boldsymbol{\theta})$, where f is a convex function over a non-empty convex set $\mathcal{S} = \{\boldsymbol{\theta} \mid c_n(\boldsymbol{\theta}) \geq 0, n = 1, \ldots, N\}$ of *feasible solutions*. This problem can be solved using a so called barrier function (e.g. Bertsekas, 1995; Cominetti & Dussault, 1994; Mosheyev & Zibulevsky, 1999; Censor & Zenios, 1997), the exponential barrier being a particularly useful choice for our purposes,

$$E_\beta(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + \beta \sum_{n=1}^{N} \exp\left(-\frac{c_n(\boldsymbol{\theta})}{\beta}\right), \tag{18}$$

$\beta > 0$ being a penalty parameter. By finding a sequence of (unconstrained) minimizers $\{\boldsymbol{\theta}^t\}_t$ to (18), using any sequence $\{\beta_t\}_t$ with $\lim_{t\to\infty} \beta_t = 0$, these minimizers can be shown to converge to a global solution of the original problem, i.e. it holds:

$$\min_{\boldsymbol{\theta}\in\mathcal{S}} f(\boldsymbol{\theta}) = \lim_{\beta\to 0} \min_{\boldsymbol{\theta}} E_\beta(\theta). \tag{19}$$

The barrier minimization objective for the problem (7) using the exponential barrier can be written as:

$$\begin{aligned} E_\beta(\boldsymbol{\alpha}, b, \varepsilon, \boldsymbol{\xi}, \boldsymbol{\xi}^*) = \sum_{j=1}^{J} |\alpha_j| + C\frac{1}{N}\left(\sum_{n=1}^{N} \xi_n + \xi_n^*\right) + C\nu\varepsilon + \quad (20) \\ +\beta \exp(-\varepsilon/\beta) + \beta \sum_{n=1}^{N} [\exp(-\xi_n/\beta) + \exp(-\xi_n^*/\beta)] + \\ +\beta \sum_{n=1}^{N} \left[\exp\left(-\frac{\varepsilon + \xi_n - \delta_n}{\beta}\right) + \exp\left(-\frac{\varepsilon + \xi_n^* + \delta_n}{\beta}\right)\right]. \end{aligned}$$

where $\delta_n := y_n - \sum_{j=1}^{J} \alpha_j h_j(\mathbf{x}_n) - b$ and for simplicity we have omitted the constraints $\boldsymbol{\alpha} \geq 0$. The first line in (20) is the objective of (7), the second line corresponds to the constraints $\xi_n, \xi_n^*, \varepsilon \geq 0$. The last line implements the constraints $\delta_n \leq \varepsilon + \xi_n$ and $-\delta_n \leq \varepsilon + \xi_n^*$.

Note that by setting $\nabla_{\boldsymbol{\xi}} E_\beta = \mathbf{0}$ and $\nabla_{\boldsymbol{\xi}^*} E_\beta = \mathbf{0}$, we can find the minimizing *slack variables* $\boldsymbol{\xi}, \boldsymbol{\xi}^*$ of (20) for given β, $\boldsymbol{\alpha}$ and b. Thus, the problem of minimizing (20) is greatly simplified, as there are $2N$ variables less to optimize.

In this section, we propose an algorithm (cf. Algorithm 2) that – similar to the column generation approach of the last section – solves a sequence of optimization problems, the so called *restricted master problems*. In each iteration t of the algorithm, one selects a hypothesis and then solves (or approximately solves) an unconstrained optimization problem in $t + 2$ variables. These variables are the t hypothesis coefficients of the previous iterations, the bias b and the tube size ε.

The solution of the restricted master problem with respect to the *master problem*[1] is clearly suboptimal and one cannot easily apply (19). However, it is known how fast one can decrease β if the intermediate

[1] The (full) master problem has $J + 2$ variables.

Algorithm 2 The Barrier-Regression algorithm

argument: `Sample` $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \mathbf{y} = \{y_1, \ldots, y_N\}$
`Number of iterations` T, `Regularization constant` C
`Tube parameter` $\nu \in (0, 1)$
constants: $\beta_{start} > 0$
returns: `Linear combination from` $\mathcal{H}$.
function `BarReg`$(X, \mathbf{y}, T, C, \nu)$
  `Set` $\beta = \beta_{start}$
  for $n = 1, \ldots, N$ **do** $p_n = \exp((-y_n - \varepsilon)/\beta) - \exp((y_n - \varepsilon)/\beta)$; **endfor**
  for $t = 1, \ldots, T$
    $h_t := L(X, \mathbf{p})$
\+ $[\boldsymbol{\alpha}, b, \varepsilon] := \underset{\boldsymbol{\alpha} \geq \mathbf{0}, b, \varepsilon}{\operatorname{argmin}} \; E_\beta(\boldsymbol{\alpha}, b, \varepsilon, \boldsymbol{\xi}(\beta), \boldsymbol{\xi}^*(\beta))$
    for $n = 1, \ldots, N$ **do**
      $\delta_n := \sum_{q=1}^{t} \alpha_q h_q(\mathbf{x}_n) + b - y_n$
      $p_n := \exp\left((\delta_n - \xi_n(\beta) - \varepsilon)/\beta\right) - \exp\left((-\delta_n - \xi_n^*(\beta) - \varepsilon)/\beta\right)$
    endfor
*   **if** $\sum_n p_n h_t(\mathbf{x}_n) - 1 < \beta$, **do** $\beta := \operatorname{next}(\beta)$; **endif**
  endfor
  return $f = b + \sum_{t=1}^{J} \alpha_t h_t$
end

solutions are suboptimal (cf. Proposition 1 in Cominetti & Dussault, 1994; Rätsch et al., 2000): Roughly speaking one has to ensure that $\beta \to 0$ and $\beta \approx \|\nabla E_\beta\|$ to achieve the desired convergence in the sense of (19), where the gradient is taken with respect to all variables.

The base learner needs to find a hypothesis with a large edge $\sum_{n=1}^{N} p_n h(\mathbf{x}_n)$, as such hypotheses correspond to violated constraints in the dual problem. Whereas in the classification case the maximum edge is minimized, we have in regression just that all edges have to be below 1. Therefore, we define the *corrected edge* with respect to the constraint $\sum_{n=1}^{N} p_n h(\mathbf{x}_n) \leq 1$ (cf. (8)) as $\sum_{n=1}^{N} p_n h(\mathbf{x}_n) - 1$, which is positive, if the constraint is violated. We now consider the case where the base learner finds a hypothesis, which is only *δ-optimal with respect to the corrected edge*. By this we mean that it finds a hypothesis that is not much worse than the best hypothesis in $\mathcal{H}$, i.e.

$$L(X, \mathbf{p}) \in \left\{ h \;\middle|\; \sum_{n=1}^{N} p_n h(\mathbf{x}_n) - 1 \geq \delta \left(\max_{g \in \mathcal{H}} \sum_{n=1}^{N} p_n g(\mathbf{x}_n) - 1 \right) \right\}, \quad (21)$$

for some constant $\delta \in (0, 1]$. Note that the correction in the edge comes from the regularization term $\|\boldsymbol{\alpha}\|_1$. Then we get:

Lemma 8. *While running Algorithm 2 using a base learner satisfying (21), the barrier parameter β is decreased only if $\beta \geq \delta \|\nabla E_\beta\|_\infty$, where the gradient is taken with respect to all variables $\varepsilon, b, \alpha_1, \ldots \alpha_J$.*

Proof. The gradient of E_β with respect to ε and b is always zeros as they are unbounded variables in the minimization in line "+". The gradient of E_β with respect to α_j is

$$\nabla_{\alpha_j} E_\beta(\boldsymbol{\alpha}, b, \varepsilon, \boldsymbol{\xi}, \boldsymbol{\xi}^*) = 1 - \sum_{n=1}^{N} p_n h_j(\mathbf{x}_n),$$

where $p_n = \exp\left((\delta_n - \xi_n - \varepsilon)/\beta\right) - \exp\left((-\delta_n - \xi_n^* - \varepsilon)/\beta\right)$. We have two cases:

- The hypothesis is already in the restricted master problem: If $\nabla_{\alpha_j} E = 0$ (cf. line "+") we get $\alpha_j \geq 0$ or if $\nabla_{\alpha_j} E > 0$ we have $\alpha_j = 0$. Note that the case $\nabla_{\alpha_j} E = 0$ can not happen. Thus, the gradient projected on the feasible set ($\boldsymbol{\alpha} \geq \mathbf{0}$) is always zero.

- The hypothesis has not already been included: If $\nabla_{\alpha_j} E_\beta < 0$, the last constraint in (8) is violated for j and the hypothesis h_j needs to be included in the hypothesis set.

Thus, one can exploit the property (21) of the base learner to upper-bound the gradient of the master problem at the current solution. If the learner returns a hypothesis $h = L(X, \mathbf{p})$, then by (21) there does not exist another hypothesis with an edge larger than by a factor of δ^{-1}. Assume there exists a violated constraint. Then by line "$*$", β is decreased if $\delta \|\nabla_{\boldsymbol{\alpha}} E_\beta\|_\infty \leq \sum_{n=1}^N p_n h(\mathbf{x}_n) \ - 1 \leq \beta$. $\square$

Using this Lemma one gets the desired convergence property of Algorithm 2:

Theorem 9. *Assume $\mathcal{H}$ is finite and the base learner L satisfies condition (21). Then for $T \to \infty$ the output of the algorithm converges to a global solution of (7).*

Proof. Let E_β be given by (20). By Proposition 1 of (Cominetti & Dussault, 1994) (see Rätsch et al., 2000), one knows that any accumulation point of a sequence $\{\boldsymbol{\theta}^t\}_t$ satisfying $\|\nabla_{\boldsymbol{\theta}} E_{\beta_t}(\boldsymbol{\theta}^t)\|_\infty \to 0$ ($\beta_t \to 0$) is a global solution of (7). By Lemma 8 we have that β is decreased only if $\beta > \delta \|\nabla E_\beta\|_\infty$. If β is not decreased, the gradient will be reduced in a finite number of iterations such that $\delta \|\nabla E_\beta\|_\infty < \beta$. Thus $\beta \to 0$ and $\|\nabla E_\beta\|_\infty \to 0$. $\square$

Similar conditions can be used to prove the convergence of Algorithm 1 in the case of non-optimal base learners in the sense of (21).

Barrier methods have also been applied to semi-infinite programming problems. In Kaliski et al. (1999) a similar barrier algorithm using the log-barrier has been used (cf. also Mosheyev & Zibulevsky, 1999). It is future work to rigorously prove that Algorithm 2 also converges to the optimal solution when the hypothesis space is infinite.

The algorithms proposed here are incomplete without descriptions of the base hypothesis space and the base learner algorithm. In the next section, we consider choices of the hypothesis space and base learner, and how they effect the algorithms.

4.3. Choice of Hypothesis Space and Base Learner

Recall that both algorithms require the hypothesis $h_{\mathbf{p}}$ that solves or approximately solves

$$h_{\mathbf{p}} = \operatorname*{argmax}_{h \in \mathcal{H}} \sum_{i=n}^{N} p_n h(\mathbf{x}_n). \tag{22}$$

So the question is how do we solve this for different types of base learners. If the set of base learners is compact, then this maximum must exist.

4.3.1. *Kernel functions*

Suppose we wish to construct ensembles of functions that themselves are linear combinations of other functions (e.g. of kernel functions) using coefficient $\boldsymbol{\gamma}$, i.e. functions of the form $k_n(\cdot) \equiv k(\mathbf{x}_n, \cdot)$:

$$h^{\boldsymbol{\gamma}}(\mathbf{x}) := \sum_{n=1}^{N} \gamma_n k(\mathbf{x}_n, \mathbf{x}), \quad \boldsymbol{\gamma} \in \mathbb{R}^N. \tag{23}$$

The set $\{h^{\boldsymbol{\gamma}}\}$ is an infinite hypothesis set and is unbounded, if $\boldsymbol{\gamma}$ is unbounded. So, one has to restrict $\boldsymbol{\gamma}$ – here we consider bounding the ℓ_1-norm of $\boldsymbol{\gamma}$ by some constant, e.g. $\mathcal{H} := \{h^{\gamma} \mid \|\boldsymbol{\gamma}\|_1 \leq 1, \boldsymbol{\gamma} \in \mathbb{R}^N\}$. Then the problem (22) has a closed form solution: Let j^* be the maximum absolute sum of the kernel values weighted by $\mathbf{p}$:

$$j^* = \operatorname*{argmax}_{j=1,\ldots,N} \sum_{n=1}^{N} p_n k(\mathbf{x}_j, \mathbf{x}_n).$$

Then $h^{\boldsymbol{\gamma}}$ with $\boldsymbol{\gamma} = [0, \ldots, 0, \gamma_{j^*}, 0, \ldots, \mathbf{0}]$ is a solution to (22), where $\gamma_{j^*} = \operatorname{sign}\left(\sum_{n=1}^{N} p_n k(\mathbf{x}_{j^*}, \mathbf{x}_n)\right)$. This means, if we boost linear combinations of kernel functions bounded by the ℓ_1-norm of $\boldsymbol{\gamma}$, then we will be adding in exactly one kernel basis function $k(\mathbf{x}_{j^*}, \cdot)$ per iteration. The resulting problem will be exactly the same as if we were optimizing a SVM regression LP (e.g. Smola et al., 1999) in the first place. The only difference is that we have now defined an algorithm for optimizing the function by adding one kernel basis at a time. So while we posed this problem as a semi-infinite learning problem it is exactly equivalent to the finite SVM case where the set of hypotheses being boosted is the individual kernel functions $k(\mathbf{x}_n, \mathbf{x})$.

If the γ_i were bounded using different norms then this would no longer be true. We would be adding functions that were the sum of many kernel functions (for using the ℓ_2-norm, see Rätsch et al., 2000a). Likewise, if we performed an active kernel strategy, where the set of kernels is parameterized over some set then the algorithm would change. We consider this problem in the next section.

4.3.2. *Active Kernel Functions*

Now consider the case where we chose a set of (kernel) functions parameterized by some vector $\boldsymbol{\kappa}$. By the same argument above, if we impose the bound $\|\boldsymbol{\gamma}\|_1 \leq 1$, we need only consider one such basis function at a time. But in this case since the kernel is parameterized over a set of continuous values $\boldsymbol{\gamma}$, we will have an infinite set of hypothesis. Say for example we wish to pick the a RBF kernel with parameters $\boldsymbol{\mu}$ (the center) and σ^2 (the variance), i.e. $\boldsymbol{\gamma} = [\boldsymbol{\mu}, \sigma]$. Then we chose the hypothesis function ($s = \dim(\mathbf{x})$)

$$h^{(\hat{\boldsymbol{\mu}},\hat{\sigma})}(\mathbf{x}) = \frac{1}{(2\pi\hat{\sigma}^2)^{s/2}} \exp\left(-\frac{\|\mathbf{x} - \hat{\boldsymbol{\mu}}\|_2^2}{2\hat{\sigma}^2}\right)$$

with parameters $(\hat{\boldsymbol{\mu}}, \hat{\sigma})$ that maximize the correlation between weight $\mathbf{p}$ and the output (the so-called edge), i.e.

$$(\hat{\boldsymbol{\mu}}, \hat{\sigma}) = \operatorname*{argmax}_{\boldsymbol{\mu},\sigma} E(\boldsymbol{\mu}, \sigma) \quad \text{where} \quad E(\boldsymbol{\mu}, \sigma) := \sum_{n=1}^{N} p_n h^{(\boldsymbol{\mu},\sigma)}(\mathbf{x}_n) \ , \quad (24)$$

With reasonable assumptions, this is a bounded function that is in $\mathbf{p}$. Thus all of the above results for the semi-infinite case hold.

There are several ways to efficiently find $\hat{\boldsymbol{\mu}}$ and $\hat{\sigma}$. The straightforward way is to employ some standard nonlinear optimization technique to maximize (24). However, for RBF kernels with fixed variance σ^2 there is a fast and easy to implement EM-like strategy. By setting $\nabla_{\boldsymbol{\mu}} E(\boldsymbol{\mu}, \sigma) = \mathbf{0}$, we get $\boldsymbol{\mu} = \sum_{n=1}^{N} q_n \mathbf{x}_n$, where $q_n = Z p_n \exp\left(\frac{-\|\mathbf{x}-\boldsymbol{\mu}\|_2^2}{2\sigma^2}\right)$, and Z is a normalization factor such that $\sum_n q_n = 1$. By this update, we are computing the weighted center of the data, where the weights depend on $\mathbf{p}$. Note, for given vector $\mathbf{q}$, one can compute (M-step) the optimal center $\boldsymbol{\mu}$. However, $\mathbf{q}$ depends on $\boldsymbol{\mu}$ and one has to iteratively recompute $\mathbf{q}$ (E-step). The iteration can be stopped, if $\sum_n q_0 = 0$ or $\boldsymbol{\mu}$ does not change anymore. As the objective function has local minima, one may start at a random position, e.g. at a random training point.

4.3.3. *SVM Classification Functions*

Here we consider the case of using a linear combination of classification functions whose output is ± 1 to form a regression function. An example of such an algorithm is the Tree-Boost algorithm of Friedman (Friedman, 1999). For absolute error functions, Tree-Boost constructs a classification tree where the class of each point is taken to be the sign of the residual of each point, i.e. points that are overestimated are assigned to class -1 and points that are underestimated are assigned to class 1. A decision tree is constructed, then based on a projected gradient descent technique with an exact line-search, each point falling in a leaf node is assigned the mean value of the dependent variables of the training data falling at that node. This corresponds to a different α_t for each node of the decision tree. So at each iteration, the virtual number of hypotheses added in some sense corresponds to the number of leaf nodes of the decision tree.

Here we will take a more simplified view and consider one node decision trees where the decision trees are linear combinations of the data. Specifically our decision function at each node is $f(\mathbf{x}, \mathbf{w}, b) = \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$. Thus at each iteration of the algorithm we want to

$$(\hat{\mathbf{w}}, \hat{b}) = \operatorname*{argmax}_{\mathbf{w},b} \left\{ \sum_{n=1}^{N} p_n \,\mathrm{sign}(\langle \mathbf{w}, \mathbf{x}_n \rangle + b) \right\} \tag{25}$$

Note that there are only finitely many ways to label N points so this is a finite set of hypotheses. There are infinitely many possible (w, b) but any that produce the same objective value are equivalent to the boosting algorithm.

The question is how to practically optimize such a problem. Clearly an upper bound on the best possible value of the above equation is obtained by any $(\mathbf{w}, b)$ solution satisfying $\mathrm{sign}(f(\mathbf{x}_n, \mathbf{w}, b)) = \mathrm{sign}(p_n)$. So in some sense, we can consider the $\mathrm{sign}(p_n)$ to be the desired class of $\mathbf{x}_n$. Now it frequently may not be possible to construct such a f. Each $\mathbf{x}_n$ that is misclassified will be penalized by exactly $|p_n|$. Thus we can think of $|p_n|$ as the misclassification cost of $\mathbf{x}_n$. Given these classes, and misclassification weights, we can use any weight sensitive classification algorithm to construct a hypothesis.

In this study we used the following problem converted into LP form to construct f:

$$\begin{array}{lll} (\hat{w}, \hat{b}) = & \underset{\mathbf{w}, b, \boldsymbol{\xi}}{\operatorname{argmin}} & \sum_{n=1}^{N} |p_n| \xi_n \\ & \text{with} & \operatorname{sign}(p_n)(\langle \mathbf{w}, \mathbf{x}_n \rangle + b) \geq 1 - \xi_n, \ n = 1, \ldots, N \\ & & \|\mathbf{w}\|_1 \leq \delta, \quad \boldsymbol{\xi} \geq 0 \end{array} \tag{26}$$

where $\delta > 0$ becomes a parameter of the problem.

Some interesting facts about this formulation. The choice of δ controls the capacity of the base learners to fit the data. For a fixed choice of δ, classification functions using a relatively fixed number of w_d nonzero. So the user can determine based on experimentation on the training data, how δ effects the complexity of the base hypothesis. Then the user may fix δ according to the desired complexity of the base hypothesis. Alternatively, a weighted variation of ν-SVMs (Schölkopf et al., 2000) could be used to dynamically chose δ.

Like in TreeBoost, we would like to allow each side of the linear decision to have a different weight. We describe the changes required to Algorithm 1 to allow this. At each iteration, LP (26) is solved to find a candidate hypothesis $(\hat{\mathbf{w}}, \hat{b})$. Then instead of adding a single column to the restricted master LP (12), two columns are added. The first column is $h_{t_+} = I(\langle \hat{\mathbf{w}}, \mathbf{x} \rangle + b > 0)$ and the second column is $h_{t_-} = -I(\langle \hat{\mathbf{w}}, \mathbf{x} \rangle + b < 0)$. The algorithms stop if both of these hypotheses do not meet the criteria given in the algorithm. The algorithm should terminate if $\sum_n h_{t_+}(\mathbf{x}_n) + h_{t_-}(\mathbf{x}_n)(\mathbf{d}_n - \mathbf{d}_n^*) < 2$. We call this variant of the algorithm CG-LP. This change has no effect on the convergence properties.

5. Experiments

In this section we present some preliminary results indicating the feasibility of our approaches. We will start in Section 5.1 with showing some basic properties of the CG and barrier algorithms for regression. We show that both algorithms are able to produce excellent fits on a noiseless and several noisy toy problems.

As base learners we use the three proposed in Section 4.3. We will denote by CG-k, CG-ak and CG-LP, the CG algorithms using RBF kernels, active RBF kernels and classification functions as base learners, respectively. Likewise for Bar-k, Bar-ak and Bar-LP using the barrier algorithm. Not all of these possible combinations have been implemented.

To show the competitiveness of our algorithms we performed a benchmark comparison in Section 5.2 on time-series prediction problems that have been extensively studied in the past.

Moreover, we give an interesting application to a problem derived from computer-aided drug-design in Section 5.3. There, we in particular show that the approach using classification functions as base learner is very well suited for datasets where the dimensionality of the problem is high, but the number of samples is very small.

5.1. An Experiment on toy data

To illustrate (i) that the proposed regression algorithm converges to the optimal (i.e. zero error) solution and (ii) is capable of finding a good fit to noisy data (signal:noise=2:1) we applied it to a toy example – the frequently used sinc function ($\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$) in the range $[-2\pi, 2\pi]$. For our demonstration (cf. Fig. 1) we used two base hypothesis spaces: (i) RBF kernels in the way described in Section 4.3.1, i.e.

$$\mathcal{H} = \{h_n(\mathbf{x}) = \exp(-\|\mathbf{x} - \mathbf{x}_n\|^2/\sigma^2) \mid n = 1, \ldots, N\}$$

with $\sigma^2 = 1/2$ and (ii) classification functions as described in Section 4.3.3. In the first case we used the CG and the Barrier approch – leading to the algorithms CG-k and Bar-k. The latter case is included for demonstration purposes only, the CG-LP is designed for high-dimensional data sets and does not perform well in low dimensions due to the severely restricted nature of the base hypothesis set.

To keep the results comparable between different data sets we use a normalized measure of error – the Q^2-error (also called normalized mean squared error), which is defined as:

$$Q^2 = \frac{\sum_{n=1}^{N}(y_n - f(\mathbf{x}_n))^2}{\sum_{n=1}^{N}(y_n - \frac{1}{N}\sum_i y_i)^2}. \tag{27}$$

A $Q^2 > 1$ is meaningless since simply predicting the mean target value will result in a Q^2-value of one.

Let us first consider the case of RBF-kernels. In the noise-free case (left panel of Fig. 1) we observe – as expected from Proposition 2 – that the (automatically determined) tube size ε is very small (0.0014), while it is kept large (0.12) for the high noise case (right panel). Using the right tube size, one gets an almost perfect fit ($Q^2 = 4 \cdot 10^{-5}$) in the noise-free case and an excellent fit in the noisy case ($Q^2 = 0.12$) – without re-tuning the parameters.

The CG-LP produced a piecewise-constant function based on only two classification functions. The same solution of $Q^2 = 0.4$ was produced in both the noisy and noise-free cases. Interestingly in the noisy

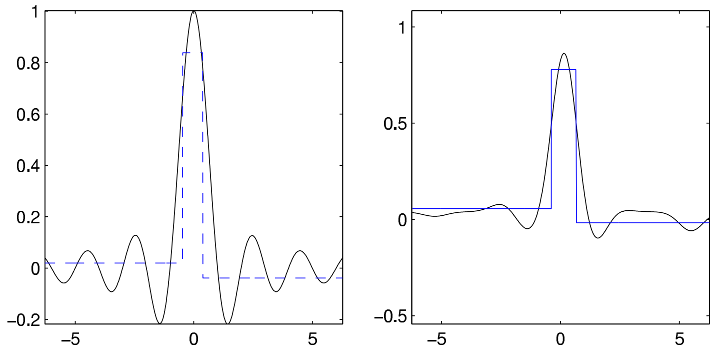

Figure 1. Toy example: The left panel shows the fit of the sinc function without noise using RBF-kernels (solid) and classification functions (dashed). The solid fit is almost perfect ($Q^2 = 4 \cdot 10^{-5}$), while the dashed function is too simple ($Q^2 = 0.4$). The right panel shows a fit using RBF-kernels ($Q^2 = 0.12$) on noisy data (signal:noise=2:1, $C = 100$). The tube size is automatically adapted by the algorithm ($\varepsilon = 0.0014$ (left) and $\varepsilon = 0.12$ (right)), such that a half of the patterns lie inside the tube ($\nu = 1/2$).

case it produces almost an identical function. Because the hypothesis space only consists of *linear* classification functions constructed by LP (26), the set of base hypothesis is extremely restricted. Thus high bias, but low variance behavior can be expected. We will see later than on high dimensional datasets the CG-LP can perform quite well.

Let us now compare the convergence speed of CG- and Barrier-Regression in the controlled setting of this toy example. For this we run both algorithms and record the objective values of the restricted master problem. In each iteration of the barrier algorithm one has to find the minimizing or almost minimizing parameters $(\boldsymbol{\alpha}, \varepsilon, b)$ of the barrier function E_β for the restricted master problem. In our implementation we use an iterative gradient descent method, where the number of gradient steps is a parameter of the algorithm. The result is shown in Fig. 2. One observes that both algorithms converge rather fast to the optimal objective value (dotted line). The CG algorithm converges faster than the barrier algorithm, as in the barrier parameter usually decreases not quick enough to compete with the very efficient Simplex method. However, if the number of gradient descent steps is large enough (e.g. 20), the barrier algorithm produces comparable results in the same number of iterations. Note that if one does only one gradient descent step per iteration, this approach is similar to the algorithm proposed in Collins et al. (2000) that uses parallel coordinate descent steps (similar to Jacobi iterations).

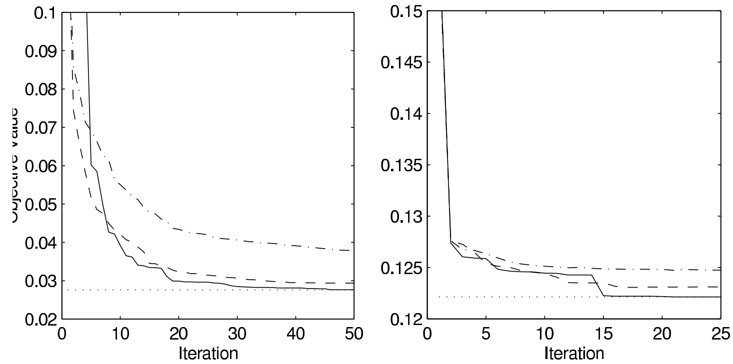

Figure 2. Convergence on the toy example: The convergence of the objective function $\|\boldsymbol{\alpha}\|_1 + \|\boldsymbol{\xi}\|_1/N + C\nu\varepsilon$ in CG-Regression (solid) and Barrier-Regression to the optimal value (dotted) over the number of iterations. Left for no noise and right for large normal noise (signal:noise=2 : 1). For Barrier-Regression we did 1 (dash-dotted) and 20 (dashed) gradient descent steps in each iteration, respectively. We used $\nu = 1/2, C = 100$ and RBF-kernels with $\sigma^2 = 1/2$. We got $\|\boldsymbol{\alpha}\|_1 = 2.7$, $\varepsilon = 0.0014$ (left) and $\|\boldsymbol{\alpha}\|_1 = 1.3$, $\varepsilon = 0.12$ (right).

5.2. Time Series Benchmarks

In this section we would like to compare our new methods to SVMs and RBF networks. For this we chose two well-known data sets that have been frequently used as benchmarks on time-series prediction: (i) the Mackey-Glass chaotic time series (Mackey & Glass, 1977) and (ii) data set D from the Santa Fe competition (Weigend & N.A. Gershenfeld (Eds.), 1994). We fix the following experimental setup for our comparison. We use seven different models for our comparison: three models that have been used in Müller et al. (1999) (RBF nets and SVM-Regression (SVR) with linear and Huber loss) and four new models: CG-k, CG-ak, Bar-k and Bar-ak.

All models are trained using a simple cross validation technique. We choose the model with the minimum prediction error measured on a randomly chosen validation set (originally taken from Müller et al., 1999). The data including our experimental results can be obtained from `http://ida.first.gmd.de/~raetsch/data/ts`.

5.2.1. *Mackey Glass Equation*

Our first application is a high-dimensional chaotic system generated by the Mackey-Glass delay differential equation

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-t_\Delta)}{1 + x(t-t_\Delta)^{10}}, \tag{28}$$

with delay $t_\Delta = 17$. Eq. (28) was originally introduced as a model of blood cell regulation (Mackey & Glass, 1977) and became quite common as an artificial forecasting benchmark. After integrating (28), we added noise to the time series. We obtained training (1000 patterns) and validation (the following 194 patterns) sets using an embedding dimension $d = 6$ and a step size $\tau = 6$. The test set (1000 patterns) is noiseless to measure the true prediction error. We conducted experiments for different signal to noise ratios[2] (SNR) using uniform noise.

In Table II we state the results given in the original paper (Müller et al., 1999) for SVMs using ε-insensitive loss and Huber's robust loss (quadratic/linear) and RBF networks. Moreover, we give the results for the CG and the barrier algorithm using RBF kernels and active RBF-kernels.[3] We also applied the CG algorithm using classification functions (CG-LP), but the algorithm performed very poorly ($Q^2 \approx 0.16$), because it could not generate complex enough functions. From Table II we observe that all four algorithms perform on average as good as the best of the other algorithms (in 11 cases better and in 13 cases worse). The 100 step prediction at low noise levels is rather poor compared to SVMs, but it is great on the higher noise levels.

Note that the CG and the barrier algorithm do not perform significantly different (CG is in 5 cases better and in 7 cases worse). This shows that the simple barrier implementation given in Algorithm 2 achieves a high enough accuracy to compete with a sophisticated simplex implementation used in the CG-algorithms.

5.2.2. *Data Set D from the Santa Fe Competition*

Data set D from the Santa Fe competition is artificial data generated from a nine-dimensional periodically driven dissipative dynamical system with an asymmetrical four-well potential and a slight drift on the parameters (Weigend & N.A. Gershenfeld (Eds.), 1994). The system has the property of operating in one well for some time and then switching to another well with a different dynamical behavior. Therefore, we first segment the time series into regimes of approximately stationary dynamics. This is accomplished by applying the *Annealed Competition of Experts* (ACE) method described in Pawelzik et al. (1996), Müller et al. (1995) (no assumption about the number of stationary subsystems was made). Moreover, in order to reduce the effect of the continuous

[2] We define the SNR in this experiment as the ratio between the variance of the noise and the variance of the data.

[3] On the entries set as italic, the model selection failed completely. In this case we selected the model manually by chosing the model on the 10th percentile of the test errors over all tested models.

drift, only the last 2000 data points of the training set are used for segmentation. After applying the ACE algorithm, the data points are individually assigned to classes of different dynamical modes. We then select the particular class of data that includes the data points at the end of Data Set D as the training set.[4]

This allows us to train our models on quasi-stationary data and we avoid having to predict the average over all dynamical modes hidden in the full training set (see also Pawelzik et al. (1996) for further discussion). However, at the same time we are left with a rather small training set requiring careful regularization, since there are only 327 patterns in the extracted training set. As in the previous section we use a validation set (50 patterns of the extracted quasi-stationary data) to determine the model parameters of SVMs, RBF networks and CG-Regression. The embedding parameters used, $d = 20$ and $\tau = 1$, are the same for all the methods compared in Table III.

Table III shows the errors (Q^2-value) for the 25 step iterated prediction.[5] In the previous result of (Müller et al., 1999) the Support vector machine with ε-ins. loss is 30% better than the one achieved by Pawelzik et al. (Pawelzik et al., 1996). This is the current record on this dataset. Given that it is quite hard to beat this record, our methods perform

[4] Hereby we assume that the class of data that generated the last points in the training set is the one that is also responsible for the first couple of steps of the iterated continuation that we aim to predict.

[5] Iterated prediction means that based on the past predictions (and not on the original data) the new prediction is computed.

Table II. 1S denotes the 1-step prediction error (Q^2) on the test set. 100S is the 100-step iterated autonomous prediction. "SNR" is the ratio between the variance of the respective noise and the underlying time series.

SNR	6.2%		12.4%		18.6%	
test error	1S	100S	1S	100S	1S	100S
CG-k	0.0011	0.0804	0.0035	0.0838	0.0031	0.0882
CG-ak	0.0010	0.0749	0.0035	0.0854	0.0065	0.0998
BAR-k	*0.0013*	*0.0900*	0.0032	0.0590	0.0051	0.0661
BAR-ak	0.0012	0.0893	0.0027	0.0621	0.0066	0.0821
SVM ε-ins.	0.0007	0.0158	0.0028	0.0988	0.0057	0.4065
SVM Huber	0.0013	0.0339	0.0038	0.0852	0.0071	1.0297
RBF-NN	0.0016	0.0775	0.0038	0.1389	0.0154	1.6032

quite well. CG-ak improves the result in Pawelzik et al. (1996) by 28%, while CG-k is 26% better.[6] This is very close to the previous result. The model-selection is a crucial issue for this benchmark competition. The model, which is selected on the basis of the best prediction on the 50 validation patterns, turns out to be rather suboptimal. Thus, more sophisticated model selection methods are needed here to obtain more reliable results.

Table III. Comparison (under competition conditions) of 25 step iterated predictions (Q^2-value) on Data set D. A prior segmentation of the data according to (Müller et al., 1995; Pawelzik et al., 1996) was done as preprocessing.

CG		SVM		Neural Net	
CG-k	CG-ak	ε-ins.	Huber	RBF	PKM
0.036	0.035	0.032	0.033	0.060	0.066

5.3. Experiments on Drug data

This data set is taken from computer-aided drug design. The goal is to predict bio-reactivity of molecules based on molecular structure through the creation of Quantitative Structure-Activity Relationship (QSAR) models. Once a predictive model has been constructed, large databases can be screened cost effectively for desirable chemical properties. Then this small subset of molecules can then be tested further using traditional laboratory techniques. The target of this dataset LCCKA is the logarithm of the concentration of each compound that is required to produce 50 percent inhibition of site "A" of the Cholecystokinin (CCK) molecule. These CCK and CCK-like molecules serve important roles as neuro-transmitters and/or neuro-modulators. 66 compounds were taken from the Merck CCK inhibitor data set. The dataset originally consisted of 323 descriptors taken from a combination of "traditional" 2D, 3D, and topological properties and electron density derived TAE (Transferable Atomic Equivalent) molecular descriptors derived using wavelets (Brenema et al., 2000). All data was scaled to be between 0 and 1. The data can be obtained from `http://www.rpi.edu/~bennek`.

It is well known that appropriate feature selection on this dataset and others is essential for good performance of QSAR models due to the

[6] We have not performed experiments with the barrier algorithm on this data, since the performance is expected to be similar.

small amount of available data with known bio-reactivity and the large number of potential descriptors, see for example (Embrechts et al., 1998). In an unrelated study (Bennett et al., 2001) feature selection was done by constructing a ℓ_1-norm linear support vector regression machine (like in equation (6) but where the features are the input dimensions) to produce a sparse weighting of the descriptors. Only the descriptors with positive weights were retained. We take the reduced set of 39 descriptors as given. We refer to the full data set as LCCKA and the reduced dataset as LCCKA-R.

The typical performance measured used to evaluate QSAR data is the average sum squared error between the predicted and true target values divided by the true target variance. This is Q^2 as defined in (27). A Q^2 of less than 0.3 is considered very good. To measure the performance, 6-fold cross validation was performed. We report the out-of-sample Q^2 averaged over the 6 folds. In this preliminary study, model-selection using parameter selection techniques was not performed. As models we consider CG-LP (CG with classification functions) and CG-k (CG with non-active kernels) described in Sections 4.3.3 and 4.3.1. For CG-k, we used only three different values for the regularization constant C, the tube-parameter ν and the parameter of the base learner σ (kernel-width) and δ (complexity parameter in (26)), respectively. Thus, we examined 27 different parameter combinations. For CG-LP, we used parameter values found to work well on a reduced dataset in Bennett et al. (2001) and then chose C and δ such that the number of hypotheses and attributes per hypothesis were similar on the training data. Research is in progress to repeat these studies using a more appropriate model selection technique – leave-one-out cross validation. Model selection is critical for performance of these methods, thus efficient model selection techniques is an important open question that needs to be addressed.

First we tried CG-k on the full data set LCCKA, but it failed to achieve good performance ($Q^2 = 0.48$), while the simple approach CG-LP performed quite well with $Q^2 = 0.33$. This is because CG-LP is able to select the discriminative features based on subsets of the attributes, while the kernel-approaches get confused by the uninformative features. For the reduced set LCCKA-R, where the features are already pre-selected, the kernel approach improves significantly ($Q^2 = 0.27$) and is not significantly different than CG-LP ($Q^2 = 0.25$). Both methods produced sparse ensembles.

On the full dataset, using parameters $C = 8$, $\nu = 0.8$, and $\delta = 6$, CG-LP used on average ensembles containing 22 hypotheses consisting of, on average, 10.1 of the possible 323 attributes, while CG-k with RBF-kernel ($\sigma = 30$) and $\nu = 0.1$ used 45 hypotheses. On the reduced dataset, using parameters $C = 15$, $\nu = 0.8$, and $\delta = 10$, CG-LP used on average ensembles containing 23.5 hypotheses consisting of, on average, 10.7 attributes, while the CG-k approach ($\sigma = 10$) used on average 30.3 hypotheses ($\nu = 0.1$). The slight difference between CG-LP and CG-k might be explained again by the presence of uninformative features.

Summarizing, the CG-LP approach seems to be a very robust method to learn simple regression functions in high-dimensional spaces with automatic feature selection.

6. Conclusion

In this work we examined an LP for constructing regression ensembles based on the ℓ_1-norm regularized ϵ-insensitive loss function used for support vector machines first proposed for ensembles of finite hypothesis sets in Smola et al. (1999). We used the dual formulation of the finite regression LP: (i) to rigorously define a proper extension to the infinite hypothesis case and (ii) to derive two efficient algorithms for solving them. It is shown theoretically and empirically that even if the hypothesis space is infinite, only a small finite set of the hypotheses is needed to express the optimal solution (cf. Corollary 4). This sparseness is possible due to the use of the ℓ_1-norm of the hypothesis coefficient vector, which acts as a sparsity-regularizer.

We proposed two different algorithms for efficiently computing optimal finite ensembles. Here, the base-learner acts as an oracle to find the constraints in the dual semi-infinite problem that are violated. For the first algorithm (the CG algorithm for regression), which is based on a simplex method, we proved the convergence for the infinite case (cf. Theorem 7). The second algorithm – the Barrier algorithm for Regression – is based on an exponential barrier method that has connections to the original AdaBoost method for classification (cf. Rätsch et al., 2000). This algorithm converges for finite hypothesis classes (cf. Theorem 9). Using recent results in the mathematical programming literature (e.g. Mosheyev & Zibulevsky, 1999; Kaliski et al., 1999) we claim that it is possible to generalize it to the infinite case. Computationally both algorithms find a provably optimal solution in a small number of iterations.

We examined three types of base learning algorithms. One, based on boosting kernel functions chosen from a finite dictionary of kernels, is an example of a finite hypothesis set. We also consider active kernel methods where the kernel basis are selected from an infinite dictionary of kernels. Finally, we consider the case using the finite set of linear classification functions constructed using an LP. This is a very limited hypothesis space that is specifically designed to work on underdetermined high-dimensional problems such as the drug design data discussed in this paper.

Our preliminary simulations on toy and real world data showed that the proposed algorithms behave very well in both finite and infinite cases. In a benchmark comparison on time-series prediction problems our algorithms perform as well as the current state of the art regression methods such as support vector machines for regression. In the case of "Data set D" of the Santa Fe competition we obtained results that are as good as the current record (by SVM) on this dataset. The LP classification-based approach worked extremely well on the high-dimensional drug design datasets, since the algorithm inherently performs feature selection essential for success on such datasets.

The primary contribution of this paper has been a theoretical and conceptual study of LP-based ensemble regression algorithms in finite and infinite hypothesis spaces. For future work we plan a more rigorous investigation of the computational aspects of our approach. One open question is how to best perform selection of the LP model parameters. Another open question involves the best algorithmic approaches for solving the semi-infinite linear program. While they work well in practice, the column generation and barrier interior-point methods described here are not the current state of the art for semi-infinite linear programming. A primal-dual interior point algorithm may perform even better both theoretically and empirically especially on very large datasets. Lastly, the ability to handle infinite hypothesis sets opens up the possibility of many other possible types of base learning algorithms.

Acknowledgments: G. Rätsch would like to thank Sebastian Mika, Klaus-R. Müller, Bob Williamson and Manfred Warmuth for valuable discussions. This work was partially funded by DFG under contracts JA 379/91, JA 379/71 and MU 987/1-1 and by the National Science Foundation under Grant No. 970923 and No. 9979860.

References

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

Bennett, K., Breneman, C., Embrechts, M., Demiriz, A., Bi, J., Sukumar, N., & Lockwood, L. (2001). Support vector machines approaches to pharmaceutical modeling. Personal Communication, manuscript in preparation.

Bennett, K., Demiriz, A., & Shawe-Taylor, J. (2000). A column generation algorithm for boosting. In *Prooceedings, 17th ICML.*

Bertoni, A., Campadelli, P., & Parodi, M. (1997). A boosting algorithm for regression. In Gerstner, W., Germond, A., Hasler, M., & Nicoud, J.-D. (Eds.), *Proceedings ICANN'97, Int. Conf. on Artificial Neural Networks*, Vol. V of *LNCS*, pp. 343–348 Berlin. Springer.

Bertsekas, D. (1995). *Nonlinear Programming.* Athena Scientific, Belmont, MA.

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

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

Brenema, C., Sukumar, N., Bennett, K., Embrechts, M., Sundling, M., & Lockwood, L. (2000). Wavelet representations of molecular electronic properties: Applications in adme, qspr, and qsar. Presentation, QSAR in Cells Symposium of the Computers in Chemistry Division's 220th American Chemistry Society National Meeting.

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

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

Collins, M., Schapire, R., & Singer, Y. (2000). Adaboost and logistic regression unified in the context of information geometry. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory.*

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

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

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

Duffy, N., & Helmbold, D. (2000). Leveraging for regression. In *Colt'00*, pp. 208–219.

Embrechts, M., Kewley, R., & Breneman, C. (1998). computationally intelligent data mining for the automated design and discovery of novel pharmaceuticals. In et al., C. D. (Ed.), *Intelligent Engineering Systems Through Artifical Neural Networks*, pp. 397–403152–161. ASME Press.

Fisher, Jr., D. H. (Ed.). (1997). *Improving regressors using boosting techniques*, Proceedings of the Fourteenth International Conference on Machine Learning.

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

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

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

Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pp. 148–146. Morgan Kaufmann.

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

Friedman, J. (1999). Greedy function approximation. Technical report, Department of Statistics, Stanford University.

Frisch, K. (1955). The logarithmic potential method of convex programming. Memorandum, University Institute of Economics, Oslo.

G. Ridgeway, D. Madigan, T. R. (1999). Boosting methodology for regression problems. In Heckerman, D., & Whittaker, J. (Eds.), *Proceedings of Artificial Intelligence and Statistics '99*, pp. 152–161.

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

Hettich, R., & Kortanek, K. (1993). Semi-infinite programming: Theory, methods and applications. *SIAM Review*, *3*, 380–429.

Kaliski, J., Haglin, D., C.Roos, & Terlaky, T. (1999). Logarithmic barrier decomposition methods for semi-infinite programming. Submitted to Elsevier Science.

Kivinen, J., & Warmuth, M. (1999). Boosting as entropy projection. In *Proc. 12th Annual Conference on Computational Learning Theory*, pp. 134–144. ACM Press.

LeCun, Y., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Müller, U. A., Säckinger, E., Simard, P., & Vapnik, V. (1995). Comparison of learning algorithms for handwritten digit recognition. In Fogelman-Soulié, F., & Gallinari, P. (Eds.), *Proceedings ICANN'95 — International Conference on Artificial Neural Networks*, Vol. II, pp. 53 – 60 Nanterre, France. EC2.

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

Mackey, M. C., & Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science*, *197*, 287–289.

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

Mason, L., Bartlett, P., & Baxter, J. (1998). Improved generalization through explicit optimization of margins. Tech. rep., Deparment of Systems Engineering, Australian National University.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (1999). Functional gradient techniques for combining hypotheses. In Smola, A., Bartlett, P., Schölkopf, B., & Schuurmans, D. (Eds.), *Advances in Large Margin Classifiers*, pp. 221–247. MIT Press, Cambridge, MA.

Mika, S., Rätsch, G., & Müller, K.-R. (2001). A mathematical programming approach to the Kernel Fisher algorithm. In *Advances in Neural Information Processing Systems 13*. In press.

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

Müller, K.-R., Kohlmorgen, J., & Pawelzik, K. (1995). Analysis of switching dynamics with competing neural networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *E78–A*(10), 1306–1315.

Müller, K.-R., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., & Vapnik, V. (1999). Predicting time series with support vector machines. In Schölkopf, B., Burges, C. J. C., & Smola, A. J. (Eds.), *Advances in Kernel Methods — Support Vector Learning*, pp. 243–254 Cambridge, MA. MIT Press. Short version appeared in ICANN'97, Springer Lecture Notes in Computer Science.

Müller, K.-R., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., & Vapnik, V. (1997). Predicting time series with support vector machines. In Gerstner, W., Germond, A., Hasler, M., & Nicoud, J.-D. (Eds.), *Artificial Neural Networks — ICANN'97*, pp. 999 – 1004 Berlin. Springer Lecture Notes in Computer Science, Vol. 1327.

Pawelzik, K., Kohlmorgen, J., & Müller, K.-R. (1996). Annealed competition of experts for a segmentation and classification of switching dynamics. *Neural Computation*, *8*(2), 342–358.

Rätsch, G. (2001). *Regularized Sparse Ensemble Learning*. Ph.D. thesis, University of Potsdam, Neues Palais 10, 14469 Potsdam, Germany. in preparation.

Rätsch, G., Onoda, T., & Müller, K.-R. (2001). Soft margins for AdaBoost. *Machine Learning*, *42*(3), 287–320. also NeuroCOLT Technical Report NC-TR-1998-021.

Rätsch, G., Schölkopf, B., Mika, S., & Müller, K.-R. (2000a). SVM and Boosting: One class. Tech. rep. 119, GMD FIRST, Berlin.

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

Rätsch, G., Warmuth, M., Mika, S., & Müller, K.-R. (2001). Barrier boosting: Boosting as an optimization technique. forthcoming.

Rätsch, G., Warmuth, M., Mika, S., Onoda, T., Lemm, S., & Müller, K.-R. (2000). Barrier boosting. In *COLT'2000*, pp. 170–179. Morgan Kaufmann.

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

Schölkopf, B., Burges, C., & Smola, A. (Eds.). (1999). *Advances in Kernel Methods – Support Vector Learning*. MIT Press.

Schölkopf, B., Smola, A., Williamson, R. C., & Bartlett, P. L. (2000). New support vector algorithms. *Neural Computation*, *12*, 1207 – 1245.

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

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

Smola, A., Schölkopf, B., & Rätsch, G. (1999). Linear programs for automatic accuracy control in regression. In *Proceedings ICANN'99, Int. Conf. on Artificial Neural Networks* Berlin. Springer.

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

Weigend, A., & N.A. Gershenfeld (Eds.) (1994). *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley. Santa Fe Institute Studies in the Sciences of Complexity.

Zemel, R., & Pitassi, T. (2001). A gradient-based boosting algorithm for regression problems. In *Advances in Neural Information Precessing Systems 13* Cambridge, MA. MIT Press. In Press.