Different Boosting Algorithms and Underlying Optimization Problems

Maya Hristakeva

March 21, 2008

1 Introduction

Boosting is an ensemble based method which attempts to boost the accuracy of any given learning algorithm by applying it several times on slightly modified training data and then combining the results in a suitable manner. The boosting algorithms that we covered in class were AdaBoost, LPBoost, TotalBoost, SoftBoost, and Entropy Regularized LPBoost. The basic idea behind these boosting algorithms is that at each iteration, a weak learner learns the training data with respect to a distribution. The weak learner is then added to the final strong learner. This is typically done by weighting the weak learner in some manner, which is typically related to the weak learner's accuracy. After the weak learner is added to the final strong learner, the data is reweighted: examples that are misclassified gain weight and examples that are classified correctly lose weight. Thus, future weak learners will focus more on the examples that previous weak learners misclassified.

In this project we implemented all of the above mentioned boosting algorithms (AdaBoost, LPBoost, TotalBoost, SoftBoost, and Entropy Regularized LPBoost), as well as a new boosting algorithm in which the update is motivated by a softmax function over all hypothesis edges and a relative entropy term as a regularizer. In this project, we refer to this new boosting algorithm as Symmetric SoftmaxBoost. It has performance comparable to LPBoost and Entropy Regularized LPBoost. It starts as quickly as LPBoost and Entropy Regularized LPBoost, and it levels off as it reaches the softmax over the edges. Moreover, Symmetric SoftMaxBoost quickly reaches low generalization error and does not overfit the data. It takes fewer iterations than TotalBoost and SoftBoost in reaching low generalization error, and it is competitive with LPBoost and Entropy Regularized LPBoost. However, we do not know of any iteration bound for Symmetric SoftMaxBoost or whether it will perform as well on other types of data.

2 Project Goals

The main goals of this project were to get good understanding of the underlying theory and implementations of the boosting algorithms covered in class, and to implement a new boosting algorithm which combined a softmax function over the hypothesis edges and a relative entropy regularizer. In particular, we implemented AdaBoost, LPBoost, TotalBoost, SoftBoost, and Entropy Regularized LPBoost and compared the Symmetric SoftmaxBoost against them.

In order to implement all of these booosting algorithms (with the exception of AdaBoost), we not only had to learn certain techniques for solving their underlying optimization problems, but also had to understand how to represent them in the Matlab optimization library. This turned out

to be quite a challenging task, ,and we thank Karen Glocer for her continuous assistance regarding implementation issues. We ran into some precision issues in Matlab for the Entropy Regularized LPBoost, and thus we could not perform many experiments for different η values.

3 Boosting Setting

In the boosting setting, a boosting algorithm takes as input a set of N training examples $(x_1, y_1), ..., (x_n, y_n)$, where each instance x_i belongs to some domain X, and each label $y_i \in \pm 1$. The algorithm also maintains a distribution (i.e. a set of weights) $\mathbf{d}^t \in [0,1]^m$ on the examples. Initially, all weights are set to the uniform distribution. In each iteration $t = 1, 2, 3, ...$, a weak learner provides a new base hypothesis h_t using the current weighting of the examples based on $\mathbf{d}^t$, and then the distribution $\mathbf{d}^t$ is updated to $\mathbf{d}^{t+1}$. In the updated distribution the weights of incorrectly misclassified examples are increased, and thus, in the next iteration the weak learner is forced to focus on the remaining "hard" examples in the training set.

We can use the edge of the base hypothesis as a measure of its performance with respect to the current distribution $\mathbf{d}^t$. The edge is defined as $\sum_{n=1}^{N} d_n y_n h(x_n)$, and ideally we want it to be close to 1 (i.e. hypothesis predicts perfectly). Thus, higher edge values are associated with more useful hypothesis for classifying the training examples. The edge of a set of hypothesis is defined as the maximum edge in the set.

The final output of the boosting algorithm is always a convex combination of weak hypothesis $f_\alpha(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$, where h_t is the hypothesis added at iteration t, and α_t is its coefficient. The margin of an example (x_n, y_n) is defined as $\rho_n = y_n f_\alpha(x_n)$. This is also called the *hard margin*, and it measures by how much an example is on the correct side of the hyperplane defined by f_α. When the training examples cannot be separated by a linear combination combination of the base hypothesis, we can relax the margin constraint by replacing the hard margin with a *soft margin*. This allows for some examples to lie below the margin but they are penalized linearly via slack variables.

Boosting algorithms can also be corrective or totally corrective depending on the way they update the distributions over the examples. Corrective algorithms update the current distribution by placing a constraint only on the most recent hypothesis, where in totally corrective updates, the current distribution is constrained to have a small edge w.r.t. all of the previous hypothesis. The algorithms developed in this project used totally corrective updates based on [4, 6].

4 LPBoost

We implemented the hard and soft margin totally corrective version of LPBoost as proposed in [4, 3]. Although, there is no known logarithmic iteration bound for LPBoost, it performs very well on noisy data and this is one of the reasons we decided to explore it further. In the hard margin case, for a given set of $\{h^1, ..., h^t\}$, the basic linear programming problem that defines LPBoost is to predict with any distribution that minimizes the maximum edge of the t hypothesis seen so far.

$$\mathbf{d}^t \in \underset{\mathbf{d} \in S^N}{\operatorname{argmin}} \max_{t=1,...,T} \mathbf{u}^t \cdot \mathbf{d} \tag{1}$$

where the vector $\mathbf{u}^t$ is defined as $y_n h_t(\mathbf{x})$. It combines the current base hypothesis h_t with the true labels y_n of the N examples. Moreover, by duality the minimum maximum edge of the examples w.r.t. the hypothesis equals the maximum minimum margin:

$$\gamma^* = \min_{\mathbf{d}\in S^N} \max_{t=1,\ldots,T} \mathbf{u}^t \cdot \mathbf{d} = \max_{\alpha} \min_{n=1,\ldots,N} y_n f_\alpha(x_n) = \rho^*$$

These optimization problems are defined for the hard margin case, when we assume that the examples are linearly separable. In the soft margin case, we assume that the data is not linearly separable and we allow for some examples to lie below the margin penalizing them via slack variables ψ_n. The resulting optimization problem (2) again maximizes the minimum soft margin, and the resulting dual problem (3) minimizes the maximum edge where the distribution is capped by $1/\nu$ for $\nu \in [1, N]$. That is

$$\max_{\substack{\boldsymbol{\alpha}\in S^t \\ \psi\geq 0}} \min_{n=1,\ldots,N} (\sum_{t=1}^{T} u_n^t \alpha_t + \psi_n) - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n \tag{2}$$

$$\mathbf{d}^t \in \min_{\substack{\mathbf{d}\in S^N \\ \mathbf{d}\leq \frac{1}{\nu}\mathbf{1}}} \max_{t=1,\ldots,T} \mathbf{u}^t \cdot \mathbf{d} \tag{3}$$

LPBoost with soft margin is one of the most straightforward boosting algorithm for maximizing the soft margin. By constraining the weight on the examples, it does well on inseparable data.

Algorithm 1 Totally Corrective LPBoost with Hard Margin

Input: $S = \langle (x_1, y_1), ..., (x_n, y_n) \rangle$ and accuracy parameter $\epsilon > 0$
Initialize: $\mathbf{d^1}$ to the uniform distribution and $\hat{\gamma}_1$ to 1
Do for $t = 1, ...$

- Send $\mathbf{d^t}$ to weak learner and obtain hypothesis h^t.
  Set $u_n^t = h^t(x_n) y_n$.
- Set $\hat{\gamma}_{t+1} = \min(\hat{\gamma}_t, \mathbf{d}^t \cdot \mathbf{u}^t)$
- Update the distribution to any $\mathbf{d^{t+1}}$ that solves the LP problem

  $[\mathbf{d}^{t+1}, \gamma_t^*] \in \min_{\mathbf{d},\gamma} \gamma$ s.t. $\mathbf{d} \cdot \mathbf{u}^m \leq \gamma$, for $1 \leq m \leq t$; $\mathbf{d} \in P^N$
- **If** $\gamma_t^* \geq \hat{\gamma}_{t+1} - \epsilon$ **then** set $T = t$ and break.

Output: $f_\alpha(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$, where the coefficients α_t realize the margin γ_T^*.

Algorithm 2 Totally Corrective LPBoost with Soft Margin

Input: $S = \langle (x_1, y_1), ..., (x_n, y_n) \rangle$, accuracy parameter $\epsilon > 0$, and capping parameter $\nu \in [1, N]$
Initialize: $\mathbf{d^1}$ to the uniform distribution and $\hat{\gamma}_1$ to 1
Do for $t = 1, ...$

- Send $\mathbf{d^t}$ to weak learner and obtain hypothesis h^t.
  Set $u_n^t = h^t(x_n) y_n$.
- Set $\hat{\gamma}_{t+1} = \min(\hat{\gamma}_t, \mathbf{d}^t \cdot \mathbf{u}^t)$
- Update the distribution to any $\mathbf{d^{t+1}}$ that solves the LP problem

  $[\mathbf{d}^{t+1}, \gamma_t^*] \in \min_{\mathbf{d},\gamma} \gamma$ s.t. $\mathbf{d} \cdot \mathbf{u}^m \leq \gamma$, for $1 \leq m \leq t$; $\mathbf{d} \in P^N, \mathbf{d} \leq \frac{1}{\nu}\mathbf{1}$
- **If** $\gamma_t^* \geq \hat{\gamma}_{t+1} - \epsilon$ **then** set $T = t$ and break.

Output: $f_\alpha(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$, where the coefficients α_t that realize the margin γ_T^*.

5 Entropy Regularized LPBoost

Adding a relative entropy regularization to the linear objective of LPBoost (3), we get an algorithm that not only maximizes the soft margin, but also has an iteration bound that is logarithmic in the number of example [5]. To optimize the relative entropy, we used sequential quadratic approximation per Karin Glocer's notes. For a given set of $\{h^1, ..., h^t\}$, the modified optimization problem is defined as follows:

$$\min_{\substack{\mathbf{d} \in S^N \\ \mathbf{d} \leq \frac{1}{\nu}\mathbf{1}}} \max_{t=1,...,T} \mathbf{u}^t \cdot \mathbf{d} + \frac{1}{\eta} \triangle(\mathbf{d}, \mathbf{d}^1) \tag{4}$$

where the factor $1/\eta$ is a trade-off parameter between the relative entropy and the maximum edge. To prove the Entropy Regularized LPBoost iteration bound in [5] η should be at least $\frac{2}{\epsilon} \ln \frac{N}{\nu}$, where ϵ is an accuracy parameter. The Lagrange dual of (4) as derived in [5] is:

$$\max_{\boldsymbol{\alpha}} -\frac{1}{\eta} \log \sum_{i=1}^{n} d_i^1 \exp(-\eta \mathbf{u_i} \boldsymbol{\alpha}) - \frac{1}{\nu} \sum_{i=1}^{n} \psi_i$$
$$\text{s.t. } ||\boldsymbol{\alpha}||_1 = 1, \boldsymbol{\alpha} \geq 0, \boldsymbol{\psi} \geq 0$$

The addition of relative entropy term in (4) makes the objective function strictly convex, and thus, the Entropy Regularized LPBoost has a unique solution. Unlike LPBoost which predicts with any distribution that minimizes the maximum edge of the t hypothesis seen so far. Adding the relative entropy keeps the solution of the optimum distribution inside the simplex. When $\eta \to \infty$, the Entropy Regularized LPBoost becomes the totally corrective version of LPBoost with soft margin.

Algorithm 3 Entropy Regularized LPBoost

Input: $S = \langle (x_1, y_1), ..., (x_n, y_n) \rangle$, accuracy parameter $\epsilon > 0$, and capping parameter $\nu \in [1, N]$
Initialize: $\mathbf{d^1}$ to the uniform distribution
Do for $t = 1, ...$

- Send $\mathbf{d^t}$ to weak learner and obtain hypothesis h^t.
  Set $u_n^t = h^t(x_n) y_n$.
- Set $\delta^t = \min_{q=1,...,t} P^q(\mathbf{d}^q) - P^{t-1}(\mathbf{d}^t)$
- **If** $\delta^t \leq \epsilon/2$ **then** set $T = t - 1$ and break
- **Else** Update the distribution to

  $[\mathbf{d}^{t+1}, \gamma_t] \in \min_{\mathbf{d},\gamma} \gamma + \frac{1}{\eta} \triangle(\mathbf{d}, \mathbf{d}^1)$
  s.t. $\mathbf{d} \cdot \mathbf{u}^m \leq \gamma$, for $1 \leq m \leq t$; $\mathbf{d} \in P^N, \mathbf{d} \leq \frac{1}{\nu}\mathbf{1}$

Output: $f_\alpha(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$, where the coefficients α_t that maximize the soft margin over the hypothesis set using the LP problem (3).

The Entropy Regularized LPBoost computes the linear combination of the hypothesis based on (3), and it updates the distributions based on (4). Moreover, we need to set $\eta \geq \frac{2}{\epsilon} \ln \frac{N}{\nu}$ which would ensure that the values of the regularized problems are always at most $\epsilon/2$ bigger that the corresponding unregularized problems.

6 TotalBoost and SoftBoost

Both TotalBoost and SoftBoost are totally corrective algorithms, the current distribution is constrained to have a small edge w.r.t. all of the previous hypothesis. TotalBoost is a hard margin algorithm [4], whereas SoftBoost is a soft margin algorithm [3]. Unlike LPBoost which predicts with any distribution that minimizes the maximum edge of the t hypothesis seen so far, the weights update in both TotalBoost and SoftBoost is motivated by the minimum relative entropy principle of Jaynes: among the solutions satisfying some linear constraint choose the one that minimizes a relative entropy to the initial distribution $\mathbf{d^1}$. Therefore, among all the distributions $\mathbf{d}$ that are solutions to (1) for TotalBoost or (3) for SoftBoost, the algorithms choose the distribution that minimizes $\triangle(\mathbf{d}, \mathbf{d}^1)$. This ensures that the optimization problems of TotalBoost and SoftBoost have unique solutions. TotalBoost solves the following optimization problem:

$$\mathbf{d}^{t+1} = \min_{\mathbf{d}} \triangle(\mathbf{d}, \mathbf{d}^1) \tag{5}$$

$$\text{s.t. } \mathbf{d} \cdot \mathbf{u}^m \leq \hat{\gamma}_t - \epsilon, \text{ for } 1 \leq m \leq t; \mathbf{d} \in P^N$$

where $\hat{\gamma}_t = \min_{m=1,...,t} \mathbf{d}^m \cdot \mathbf{u}^m$.

SoftBoost algorithm is the soft margin version of TotalBoost (i.e. it adds capping to the TotalBoost algorithm). The corresponding optimization problem is defined as follows:

$$\mathbf{d}^{t+1} = \min_{\mathbf{d}} \triangle(\mathbf{d}, \mathbf{d}^1) \tag{6}$$

$$\text{s.t. } \mathbf{d} \cdot \mathbf{u}^m \leq \hat{\gamma}_t - \epsilon, \text{ for } 1 \leq m \leq t; \mathbf{d} \in P^N, \mathbf{d} \leq \frac{1}{\nu}\mathbf{1}$$

To solve the optimization problems for (5) and (6), we again used sequential quadratic programming as described in [4]. Removing the relative entropy from the objective, we arrive at the optimization problem of LPBoost. The relative entropy term in the objective function ensures that more weight is put on the examples with low margin (i.e. the ones that are hard to classify).

Algorithm 4 TotalBoost

Input: $S = \langle (x_1, y_1), ..., (x_n, y_n) \rangle$ and accuracy parameter $\epsilon < 0$
Initialize: $\mathbf{d^1}$ to the uniform distribution and $\hat{\gamma}_1$ to 1
Do for $t = 1, ...$

- Send $\mathbf{d^t}$ to weak learner and obtain hypothesis h^t.
  Set $u_n^t = h^t(x_n)y_n$.
- Set $\hat{\gamma}_{t+1} = \min(\hat{\gamma}_t, \mathbf{d}^t \cdot \mathbf{u}^t)$
- Update the distribution to

  $$\mathbf{d}^{t+1} = \underset{\mathbf{d}}{\operatorname{argmin}} \triangle(\mathbf{d}, \mathbf{d}^1) \text{ s.t. } \mathbf{d} \cdot \mathbf{u}^m \leq \hat{\gamma}_{t+1} - \epsilon, \text{ for } 1 \leq m \leq t; \mathbf{d} \in P^N$$

- **If** above problem is infeasible or $\mathbf{d^{t+1}}$ contains a zero **then** $T = t$ and break.

Output: $f_\alpha(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$, where the coefficients α_t maximize the hard margin over the hypothesis using LP problem (1).

Algorithm 5 SoftBoost

Input: $S = \langle (x_1, y_1), ..., (x_n, y_n) \rangle$ and accuracy parameter $\epsilon > 0$
Initialize: $\mathbf{d^1}$ to the uniform distribution and $\hat{\gamma}_1$ to 1
Do for $t = 1, ...$

- Send $\mathbf{d^t}$ to weak learner and obtain hypothesis h^t.
  Set $u_n^t = h^t(x_n)y_n$.
- Set $\hat{\gamma}_{t+1} = \min(\hat{\gamma}_t, \mathbf{d}^t \cdot \mathbf{u}^t)$
- Update the distribution to

  $$\mathbf{d}^{t+1} = \underset{\mathbf{d}}{\operatorname{argmin}} \triangle(\mathbf{d}, \mathbf{d}^1) \text{ s.t. } \mathbf{d} \cdot \mathbf{u}^m \leq \hat{\gamma}_{t+1} - \epsilon, \text{ for } 1 \leq m \leq t; \mathbf{d} \in P^N, \mathbf{d} \leq \tfrac{1}{\nu}\mathbf{1}$$

- **If** above problem is infeasible or $\mathbf{d^{t+1}}$ contains a zero **then** $T = t$ and break.

Output: $f_\alpha(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$, where the coefficients α_t maximize the soft margin over the hypothesis using LP problem (3).

7 New Boosting Algorithm (Symmetric SoftMaxBoost)

The final boosting algorithm that we explored is based on one of the dual optimization problems discussed in [7], for which the distribution update is motivated by minimizing a relative entropy term plus a softmax function over all hypothesis edges. In this project we refer to this new algorithm Symmetric SoftMaxBoost. For a given set of $\{h^1, ..., h^t\}$, the corresponding optimization problem is defined as follows:

$$\min_{\mathbf{d}} \frac{1}{\eta} \sum_{i=1}^{n} d_i \log \frac{d_i}{d_i^1} + \gamma \log(\sum_{j=1}^{t} \exp(\gamma^{-1} \xi_j)) \tag{7}$$

$$\text{s.t.} ||\mathbf{d}||_1 = 1$$
$$\mathbf{u}^j \mathbf{d} = \xi_j$$

Unlike all of the previously discussed boosting algorithms where we always minimized the maximum edge, this algorithm minimizes the softmax over all of the edges. In other words, Symmetric SoftMaxBoost minimizes the almost maximum edge. Moreover, the softmax function is convex, thus (7) has a unique solution.

7.1 The dual optimization problem of Symmetric SoftMaxBoost

In this section we compute the dual of (7) and discuss the relationship between the primal and the dual. The Lagrangian of (7) is:

$$L(\boldsymbol{\xi}, \mathbf{d}, \boldsymbol{\alpha}, \beta) = \frac{1}{\eta} \sum_{i=1}^{n} d_i \log \frac{d_i}{d_i^1} + \gamma \log(\sum_{j=1}^{t} \exp(\gamma^{-1} \xi_j)) + \sum_{j=1}^{t} \alpha_j (\mathbf{u}^j \cdot \mathbf{d} - \xi_j) + \beta(||\mathbf{d}||_1 - 1) \tag{8}$$

Setting the derivatives to 0:

$$\begin{aligned} \frac{\partial L}{\partial \xi_j} &= \frac{\exp \xi_j / \gamma}{\sum_q \exp(\xi_q / \gamma)} = 0 \\ \frac{\partial L}{\partial d_i} &= \frac{1}{\eta}(1 + \log \frac{d_i}{d_i^1}) + \beta + \mathbf{u}_i \boldsymbol{\alpha} = 0 \end{aligned}$$

This yields $\boldsymbol{\alpha} \geq 0, ||\boldsymbol{\alpha}||_1 = 1$ and

$$\begin{aligned} \xi_j &= \gamma \log(\alpha_j Y) \qquad \text{where } Y = \textstyle\sum_q \exp(\xi_q / \gamma) \\ d_i &= d_i^1 \exp(-\eta\beta - \eta \mathbf{u}_i \boldsymbol{\alpha} - 1) \end{aligned}$$

Plugging the optimal $\mathbf{d}$ into the Lagrangian (8) simplifies to:

$$L(\boldsymbol{\xi}, \boldsymbol{\alpha}, \beta) = \frac{1}{\eta} \sum_{i=1}^{n} d_i^1 \exp(-\eta\beta - \eta \mathbf{u}_i \boldsymbol{\alpha} - 1) + \gamma \log(\sum_{j=1}^{t} \exp(\gamma^{-1} \xi_j)) - \beta - \boldsymbol{\alpha}^T \boldsymbol{\xi} \tag{9}$$

By plugging the optimal ξ in (9) and enforcing the constraint $||\alpha||_1 = 1$, the dual function simplifies to:

$$\begin{aligned} \Theta(\boldsymbol{\alpha}, \beta) &= \frac{1}{\eta} \sum_{i=1}^{n} d_i^1 \exp(-\eta\beta - \eta \mathbf{u}_i \boldsymbol{\alpha} - 1) + \gamma \log(\sum_{j=1}^{t} \alpha_j Y)) - \beta - \gamma \sum_{j=1}^{t} \alpha_j \log(\alpha_j Y) \\ &= \frac{1}{\eta} \sum_{i=1}^{n} d_i^1 \exp(-\eta\beta - \eta \mathbf{u}_i \boldsymbol{\alpha} - 1) - \beta - \gamma \sum_{j=1}^{t} \alpha_j \log \alpha_j \end{aligned} \tag{10}$$

By differentiating $\Theta(\boldsymbol{\alpha}, \beta)$ with respect to β and setting the derivative to 0, we can determine the optimal choice of β:

$$\begin{aligned} \frac{\partial \Theta}{\partial \beta} &= -1 + \sum_{i=1}^{n} d_i^1 \exp(-\eta\beta - \eta \mathbf{u}_i \boldsymbol{\alpha} - 1) = 0 \\ \beta &= \frac{1}{\eta}(\log(\sum_{i=1}^{n} d_i^1 \exp(-\eta \mathbf{u}_i \boldsymbol{\alpha})) - 1) \end{aligned}$$

Thus, the dual problem of (10) reduces to:

$$\Theta(\boldsymbol{\alpha}) = -\frac{1}{\eta}\log\sum_{i=1}^{n} d_i^1 \exp(-\eta/mathbf{u}_i\boldsymbol{\alpha}) - \gamma\sum_{j=1}^{t}\alpha_j\log(\alpha_j) \tag{11}$$

The resulting optimization problem (11) maximizes the softmin over the margins of all examples. It consists of a softmin function plus a relative entropy term as a regularizer.

$$\max_{\boldsymbol{\alpha}} -\frac{1}{\eta}\log\sum_{i=1}^{n} d_i^1 \exp(-\eta\mathbf{u}_i\boldsymbol{\alpha}) - \gamma\sum_{j=1}^{t}\alpha_j\log(\alpha_j) \tag{12}$$
$$\text{s.t. } ||\boldsymbol{\alpha}||_1 = 1, \boldsymbol{\alpha} \geq 0$$

The primal (7) and the dual (11) of this boosting algorithm are in a way symmetric, and this is why we called the algorithm Symmetric SoftMaxBoost. In the primal, we minimize the softmax over all edges, whereas in the dual we maximize the softmin over the margins of all examples.

7.2 Connection to LPBoost and Entropy Regularized LPBoost

In the primal (7), if we let $\gamma \to 0$ then we would get the maximum edge from the softmax function. Thus, the Symmetric SoftMaxBoost becomes equivalent to the Entropy Regularized LPBoost.

$$\max_j \mathbf{u}_j\mathbf{d} = \lim_{\gamma\to 0}\gamma\log(\textstyle\sum_{j=1}^{t}\exp(\gamma^{-1}\xi_j))$$

Moreover, if we let $\gamma \to 0$ and $\eta \to \infty$ then the Symmetric SoftMaxBoost becomes the totally corrective version of LPBoost with hard margin.

In the dual domain (11), if $\eta \to \infty$ then we would get the minimum margin from the softmin function. This is equivalent to minimizing the softmax over the edges without relative entropy regularizer in the primal domain.

$$\min_i u_i\boldsymbol{\alpha} = -\lim_{\eta\to\infty}\tfrac{1}{\eta}\log(\textstyle\sum_{i=1}^{n} d_i^1\exp(-\eta\mathbf{u}_i\boldsymbol{\alpha}))$$

The Symmetric SoftMaxBoost has nice symmetries and this is why we have explored it in this project per the recommendation of Professor Manfred Warmuth.

7.3 Algorithm

The algorithm that we used for the Symmetric SoftMaxBoost is based on the previously discussed boosting algorithms. Moreover, as a result of the lack of time, we were unable to figure out a good stopping criterion for this algorithm. Thus, we run it for a set number of iterations and as long as the optimization problem is feasible.

Algorithm 6 Symmetric SoftMaxBoost

Input: $S = \langle (x_1, y_1), ..., (x_n, y_n) \rangle$ with parameters η and γ
Initialize: $\mathbf{d^1}$ to the uniform distribution
Do for $t = 1, ..., T$

- Send $\mathbf{d^t}$ to weak learner and obtain hypothesis h^t.
  Set $u_n^t = h^t(x_n)y_n$.
- Update the distribution to

  $\mathbf{d}^{t+1} = \min_{\mathbf{d}} \frac{1}{\eta} \sum_{i=1}^{n} d_i \log \frac{d_i}{d_i^1} + \gamma \log(\sum_{j=1}^{t} \exp(\gamma^{-1} \xi_j))$
  s.t. $||\mathbf{d}||_1 = 1, \mathbf{u}^t \mathbf{d} = \xi_t$
- **If** above problem is infeasible **then** $T = t$ and break.

Output: $f_\alpha(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$, where the coefficients α_t maximize the hard margin over the hypothesis using LP problem (1).

8 Experiments

We used two types of datasets in our experiments: a real one and simulated data. The first dataset was the sonar dataset from the UCI benchmark repository. It contains information of 208 objects and has 60 attributes, where the objects are classified in two classes: "rock" and "mine". We also ran the algorithms on simulated data using the model in [8]:

$$P(Y = 1|x) = q + (1 - 2q)\mathrm{I}\left[\sum_{j=1}^{J} x^{(j)} > J/2\right] \tag{13}$$

where X is distributed iid uniform on the d-dimensional unit cube $[0, 1]^d$, q is the Bayes error and $J \leq d$ is the number of effective dimensions. The constraints $n, d, J,$ and, q can be set at different values. Using this model to generate data allows us to control for the Bayes error, and thus gives us better understanding of the data.

Using the two types of datasets we separately compared the Symmetric SoftMaxBoost with the hard margin (LPBoost and TotalBoost)and soft margin (LPBoost, SoftBoost, and Entropy Regularized LPBoost) algorithms. In the reported results we used decision stumps as the base classifier, however, we also experimented with using the features as the weak learners (i.e. the hypothesis). The accuracy parameter ϵ for all of the algorithms was set to 0.01. In general, the capping parameter ν was set to two times the Bayes error ($\nu \approx 2q$) or $\nu/N = 0.1$. Finally, in the Entropy Regularized LPBoost, η was set to $\frac{2}{\epsilon} \ln \frac{N}{\nu}$, however we also experimented with some much bigger or much smaller values for η to see how the algorithm performs.

8.1 Sonar Data Results

We compared the Symmetric SoftMaxBoost against hard margin LPBoost and TotalBoost on the Sonar dataset. In Figure 1, we show the results for two different values of γ. In the figures γ is referred to as λ and Symmetric SoftMax Boost is referred to as Softmax Entropy Boost. We apologize for the inconsistency, but we did not have time to regenerate all of the figures using the new notation. In the hard margin case, Symmetric SoftMaxBoost starts as quickly as LPBoost, and it levels off as it reaches the softmax over the edges. We tested the Symmetric SoftMax Boost on different γ, and the smaller the γ was, the closer to its margin got to the LPBoost margin.

Moreover, all of the results that we report use $\frac{1}{\eta} = 0.025$ since this turned out to be a good value for the size of our datasets.

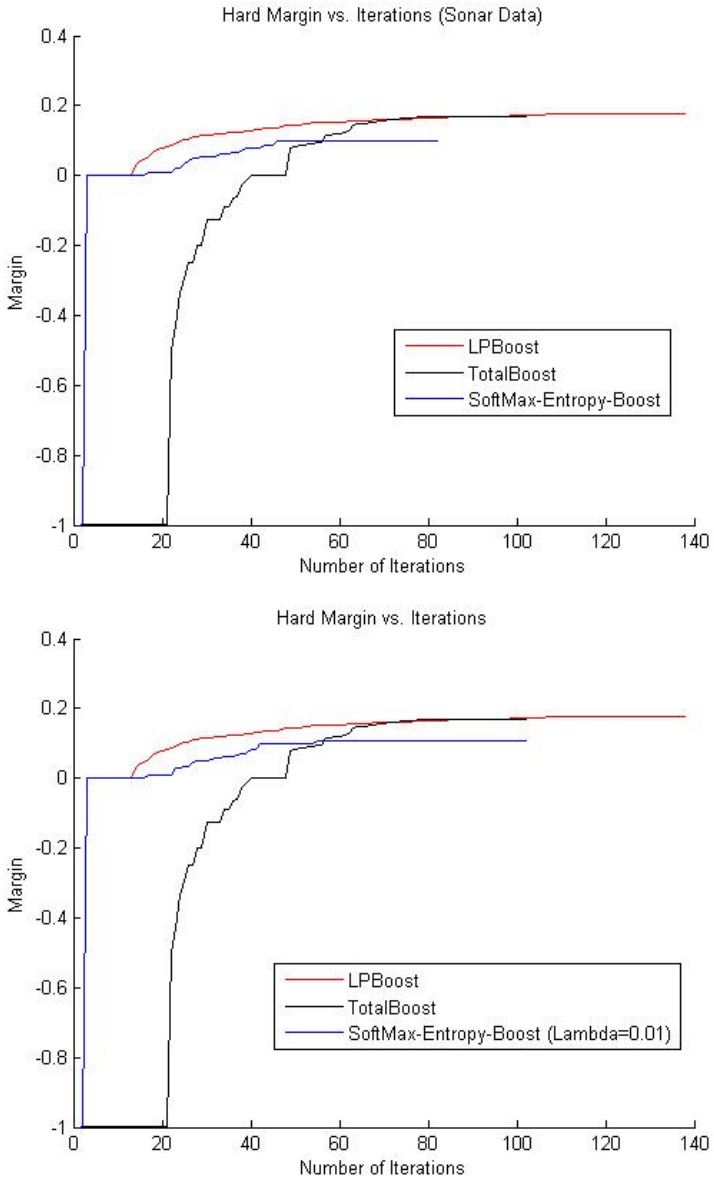

Figure 1: Comparison of Symmetric SoftMaxBoost ($\gamma = 10$ and $\gamma = 0.1$ respectively) with the hard margin LPBoost and TotalBoost on the Sonar Dataset with $\epsilon = 0.01$ and $\frac{1}{\eta} = 0.025$.

In Figure 2, we compared the Symmetric SoftMaxBoost against the soft margin LPBoost, Soft-Boost, and Entropy Regularized LPBoost again on the Sonar dataset for $\gamma = 10$and$\gamma = 0.01$. As we can see, the Symmetric SoftMaxBoost again starts as quickly as the soft margin LPBoost and the Entropy Regularized LPBoost, and then it levels off as it reaches the softmax over the edges.

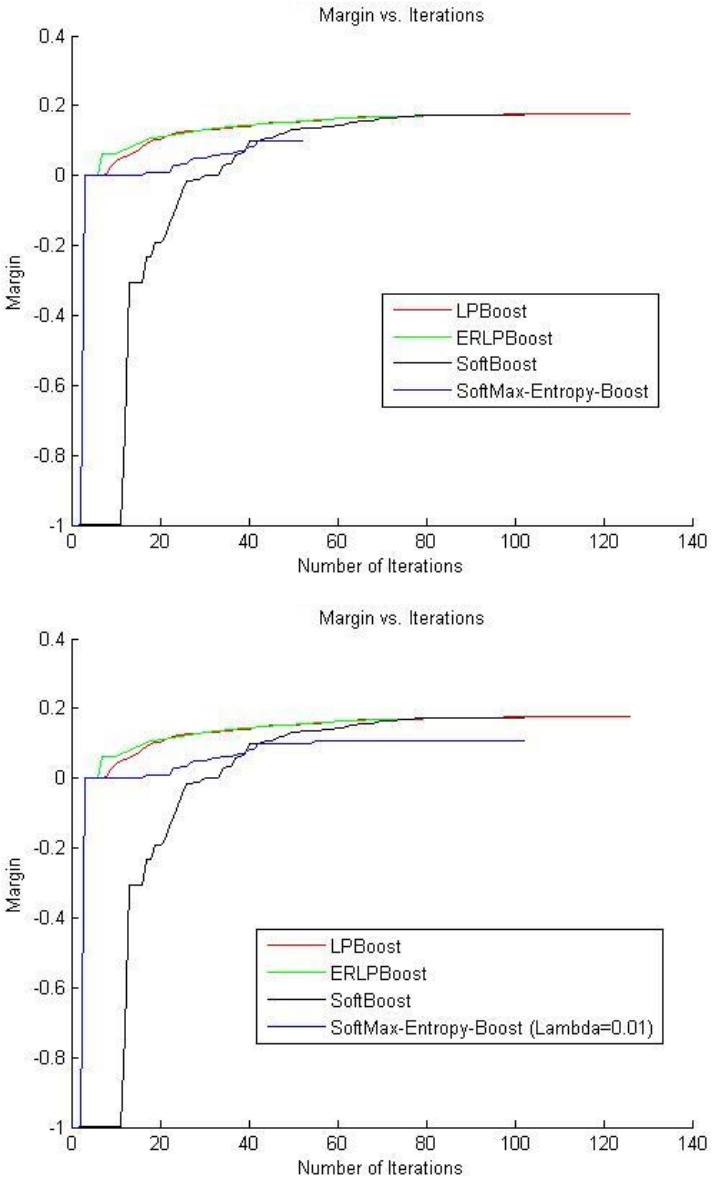

Figure 2: Comparison of Symmetric SoftMaxBoost ($\gamma = 10$ and $\gamma = 0.1$ respectively) with the soft margin LPBoost, SoftBoost, and Entropy Regularozed LPBoost on the Sonar Dataset with $\epsilon = 0.01$, $\nu/N = 0.1$, and $\eta = \frac{2}{\epsilon}\ln\frac{N}{\nu}$ (for Entropy Regularized LP Boost). Symmetric SoftMaxBoost with $\frac{1}{\eta} = 0.025$.

In Figure 3, we show the generalization error of Symmetric SoftMaxBoost against the other boosting algorithms for $\gamma = 10$. Symmetric SoftMaxBoost quickly reaches low generalization error and does not overfit the data. It takes fewer iterations than TotalBoost and SoftBoost in reaching low generalization error. It is competitive with LPBoost and Entropy Regularized LPBoost.

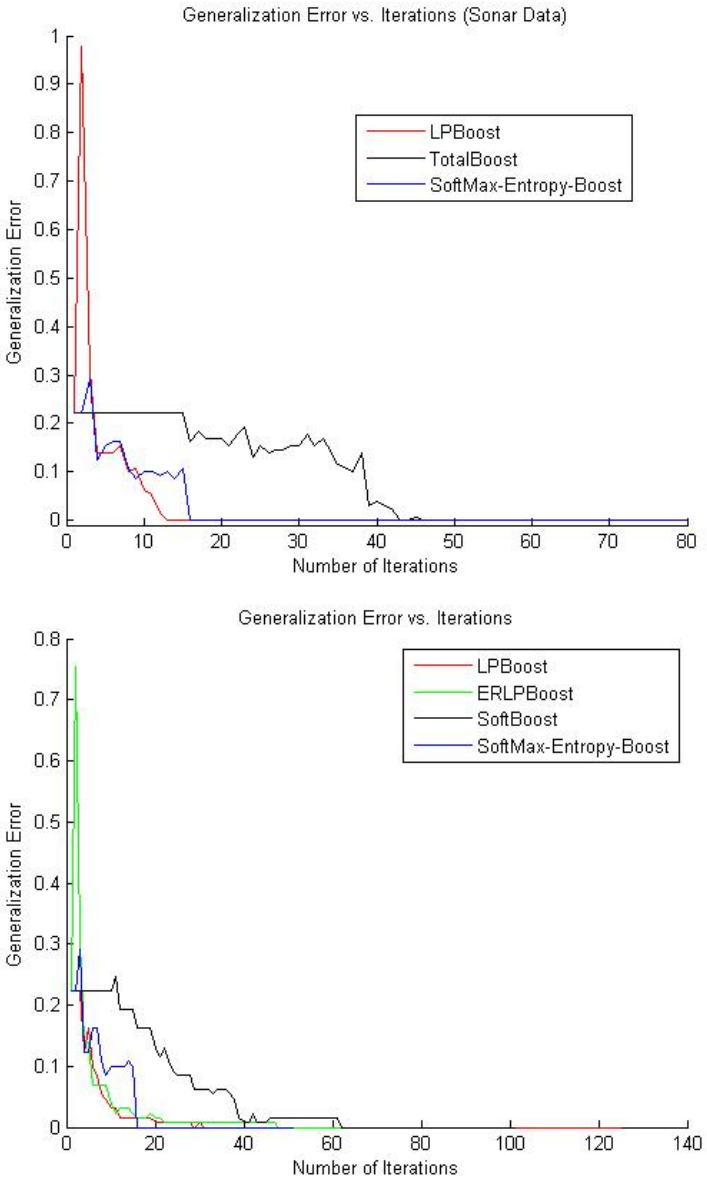

Figure 3: Generalization error of the boosting algorithms on the Sonar Dataset with $\epsilon = 0.01$, $\nu/N = 0.1$, and $\eta = \frac{2}{\epsilon}\ln\frac{N}{\nu}$ (for Entropy Regularized LP Boost). Symmetric SoftMaxBoost with $\gamma = 10$, $\frac{1}{\eta} = 0.025$.

8.2 Dataset with Bayes error $q = 0.1$

Testing the boosting algorithms on the artificially generated data using (13) we got similar results. Symmetric SoftMaxBoost starts as quickly as LPBoost and Entropy Regularized LPBoost, and it levels off as it reaches the softmax over the edges (Figure 4). Moreover, it quickly reaches low generalization error and does not overfit the data. It takes fewer iterations than TotalBoost and SoftBoost in reaching low generalization error, and it is competitive with LPBoost and Entropy Regularized LPBoost.

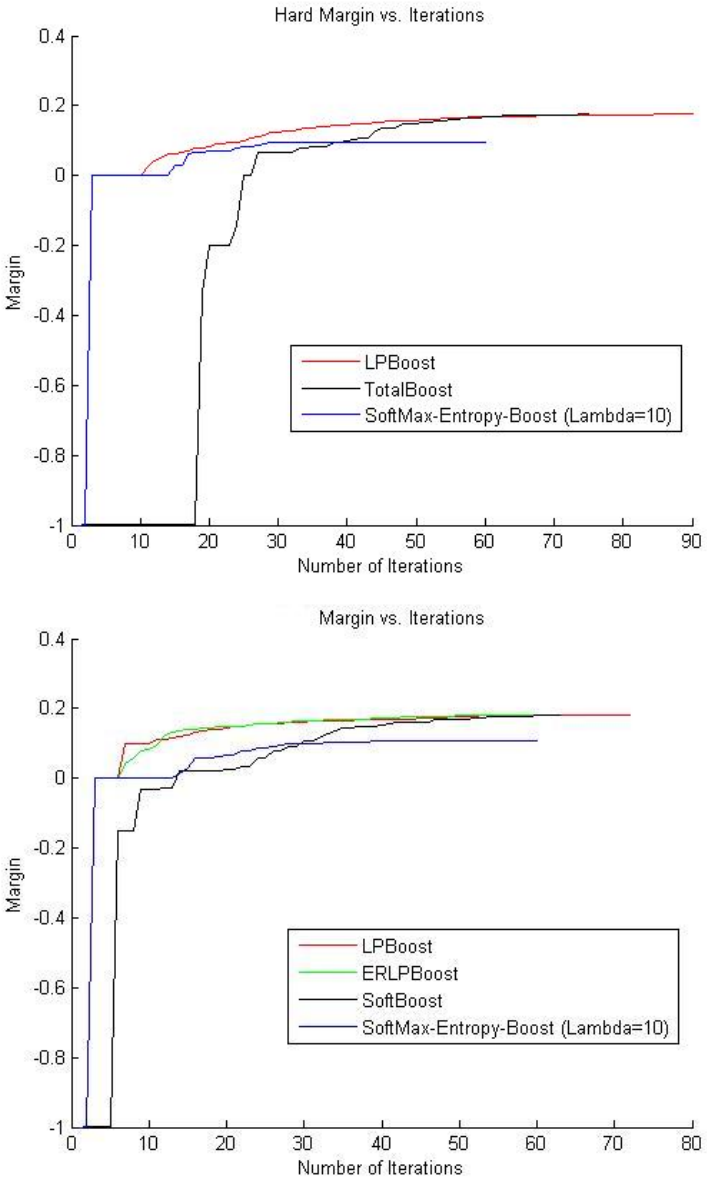

Figure 4: Data model parameters: $q = 0.1, n = 100, J = 20, d = 5$. Boosting parameters: $\epsilon = 0.01$, $\nu/N = 0.1$, and $\eta = \frac{2}{\epsilon}\ln\frac{N}{\nu}$ (for Entropy Regularized LP Boost). Symmetric SoftMaxBoost with $\gamma = 10$, $\frac{1}{\eta} = 0.025$.

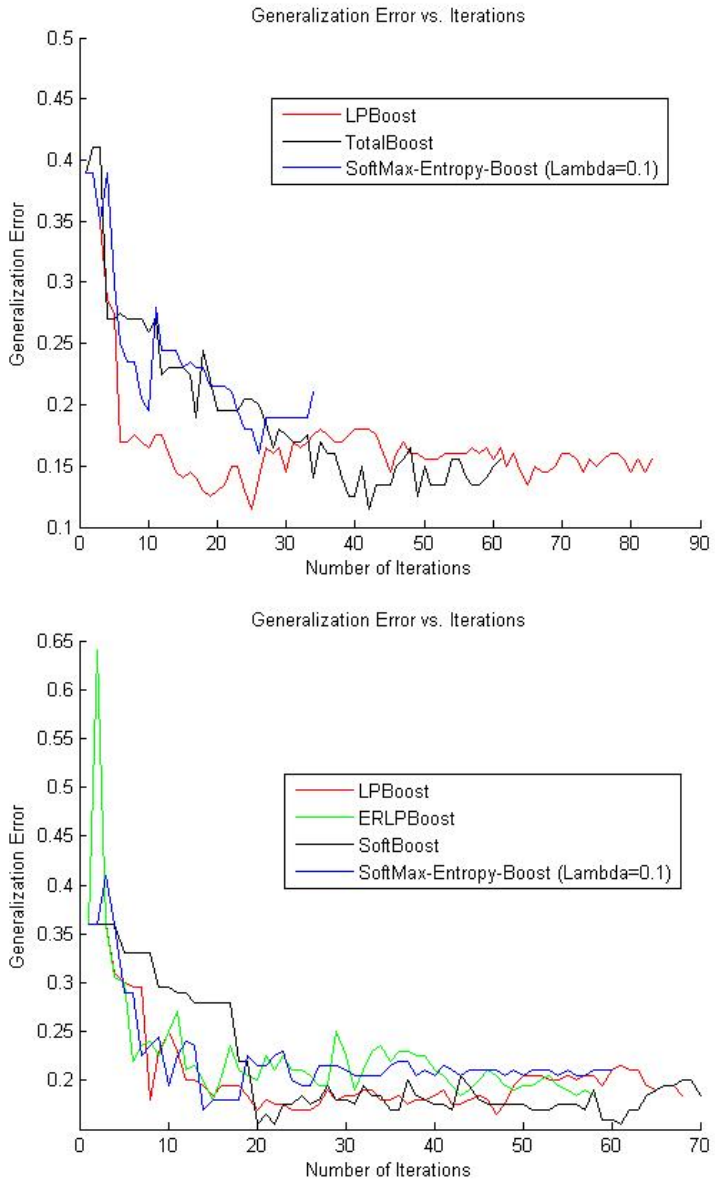

Figure 5: Data model parameters: $q = 0.1, n = 100, J = 20, d = 5$. Boosting parameters: $\epsilon = 0.01$, $\nu/N = 0.1$, and $\eta = \frac{2}{\epsilon}\ln\frac{N}{\nu}$ (for Entropy Regularized LP Boost). Symmetric SoftMaxBoost with $\lambda = 10$, $\frac{1}{\eta} = 0.025$.

8.3 Different η values for the Entropy Regularized LPBoost

We also preformed some experiments for different η values for the Entropy Regularizzed LPBoost. For the iteration bound proved in [5] to hold, eta has to be at least $\frac{2}{\epsilon}\ln\frac{N}{\nu}$. Moreover, when $\eta \to \infty$, the Entropy Regularized LPBoost becomes the totally corrective version of LPBoost with soft margin. This is supported by Figure 6, where we present the results for three different η values ($\eta = \frac{2}{\epsilon}\ln\frac{N}{\nu}$, $\eta_2 = 100(\frac{2}{\epsilon}\ln\frac{N}{\nu})$, and $\eta_3 = 0.01(\frac{2}{\epsilon}\ln\frac{N}{\nu})$).

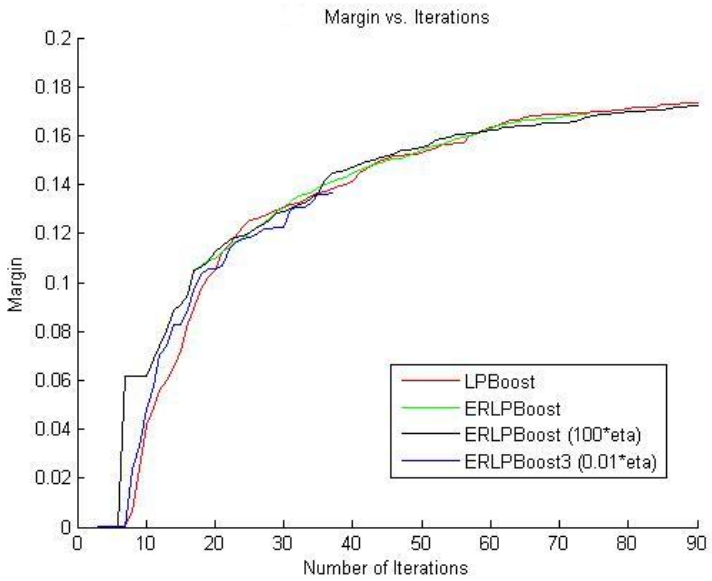

Figure 6: Entropy Regularized LPBoost with $\eta = \frac{2}{\epsilon}\ln\frac{N}{\nu}$, $\eta_2 = 100\eta$, and $\eta_3 = 0.01\eta$.

For larger values of η, Entropy Regularized LPBoost performs better than for smaller values (i.e. beats LPBoost margin more frequently).

9 Conclusion

We showed that Symmetric SoftMaxBoost has performance comparable to LPBoost and Entropy Regularized LPBoost. It starts as quickly as LPBoost and Entropy Regularized LPBoost, and it levels off as it reaches the softmax over the edges. Moreover, SoftBoost and TotalBoost start slower and take longer to reach the maximum margin. Moreover, Symmetric SoftMaxBoost quickly reaches low generalization error and does not overfit the data. It takes fewer iterations than TotalBoost and SoftBoost in reaching low generalization error, and it is competitive with LPBoost and Entropy Regularized LPBoost. However, we do not know of any iteration bound for Symmetric SoftMaxBoost or whether it will perform as well on other types of data.

Acknowledgements Thanks to Professor Warmuth for introducing me to this new boosting algorithm and for the valuable insights on why it is reasonable to use it to motivate the weights update. Moreover, many thanks to Karen Glocer for the valuable discussions and help with the implementation set-up of the optimization problems in Matlab.

References

[1] Schapire, R. (1999). A Brief Introduction to Boosting.

[2] Demeriz, A., Bennett, K.P., and Shawe-Taylor, J. (2002) Linear programming via column generation.

[3] Warmuth, M., Glocer, K., and Ratsch, G. Boosting Algorithms for Maximizing the Soft Margin. Advances in Neural Information Processing systems 20. MIT Press, 2007.

[4] Warmuth, M., Liao, J., and Ratsch, G. (2006). Totally Corrective Boosting Algorithms that Maximizing the Soft Margin. In Proc. ICML '06, pages 1001 - 1008. ACM Press, 2006.

[5] Warmuth, M., Glocer, K., and Vishwanathan, S.V.N. (2007). Entropy Regularized LPBoost.

[6] Kvinen, J., and Warmuth, M. Boosting as entropy projection. In Proc. 12th Annual conf. on computing Learning theory, pages 134-144. ACM Press, NY, 1999.

[7] Warmuth, M. and Ratsch, G. Compendium of dual optimization problems for boosting algorithms.

[8] Mease, D. and Wyner, A. Evidence Contrary to the Statistical View of Boosting. Journal of Machine Learning Research (2007).