Linear Programming Boosting by Column and Row Generation

Hatano, Kohei
Department of Informatics, Kyushu University

Takimoto, Eiji
Department of Informatics, Kyushu University

出版情報：Discovery Science (Lecture notes in computer science). 5808, pp.401-408, 2009-10-05. Springer Berlin Heidelberg
バージョン：
権利関係：

Linear Programming Boosting by Column and Row Generation

Kohei Hatano and Eiji Takimoto

Department of Informatics, Kyushu University
{hatano,eiji}@i.kyushu-u.ac.jp

Abstract. We propose a new boosting algorithm based on a linear programming formulation. Our algorithm can take advantage of the sparsity of the solution of the underlying optimization problem. In preliminary experiments, our algorithm outperforms a state-of-the-art LP solver and LPBoost especially when the solution is given by a small set of relevant hypotheses and support vectors.

1 Introduction

Learning of sparse classifiers has been popular these days in Machine Learning and related fields. For example, in text classification applications, the number of features is typically more than millions, but only a small fraction of features are likely to be relevant. In such cases, sparse classifiers are useful not only for classification but for feature selection.

A major approach for learning sparse classifiers is to formulate problems as ℓ_1 soft margin optimization, which learns a linear classifier enlarging the margin by regularizing ℓ_1 norm of the weight vector [12]. Large margin theory guarantees that this approach is robust in classification (see, e.g., [12]). Recently, the ℓ_1 soft margin optimization is also applied in learning with "similarity functions" [6, 8, 1, 14]. Since the ℓ_1 soft margin optimization is a linear program, standard optimization methods such as simplex methods or interior point methods can solve the problem. However, solving the problem directly might need much computation time even when the number of features or examples goes beyond ten thousands.

LPBoost, proposed by Demiriz et al., is a popular boosting algorithm designed to solve the soft margin optimization problem [5]. Although its iteration bound is not known and a worst case lowerbound is exponentially worse than other boosting algorithms, it is very fast in in most practical cases (earlier results of LPBoost for hard margin optimization are appeared in [7]). Given m labeled instances and n hypotheses, consider the $m \times n$ matrix in which each component is $u_{ij} = y_i h_j(\boldsymbol{x}_i)$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. Note that each row or column corresponds to an example or a hypothesis, respectively. Instead of solving the soft margin LP problem directly, LPBoost works repeatedly as follows: For each iteration t, it finds a "good" hypothesis h_t w.r.t. the current distribution

$\boldsymbol{d}_t$ and construct the next distribution d_{t+1} by solving the reduced soft margin LP problem restricted to the hypotheses set $\{h_1, \ldots, h_t\}$. The final hypothesis is given by a linear combination of past chosen hypotheses, whose coefficients are Lagrange multipliers of the reduced problem. In the the view point of the matrix, it generates columns and solve LPs repeatedly. In fact, LPBoost can be viewed as a LP solver using the column generation approach (e.g., [11]), which is a classical technique in Optimization literature.

LPBoost, however, does not seem to fully exploit the sparsity of the underlying problem. In fact, the ℓ_1-soft margin optimization problems have two kinds of sparsity. First sparsity arises in hypotheses. As explained above, only relevant hypotheses have nonzero coefficients in the optimal solution. The other sparsity appears in examples. More precisely, only some relevant examples (often called "support vectors") affect the optimal solution and the solution does not change even if other examples are removed.

In this paper, we propose a new boosting algorithm which take advantage of the sparsity of both hypotheses and examples. Our algorithm, Sparse LPBoost, takes a "column and row " generation approach. Sparse LPBoost generates seemingly relevant columns(hypotheses) and rows(examples) and solves the linear programs repeatedly. We prove that, given precision parameter $\varepsilon > 0$, Sparse LPBoost outputs the final combined hypothesis with soft margin larger than $\gamma^* - \varepsilon$, where γ^* is the optimal soft margin. Further, we propose some heuristics for choosing hypotheses and examples to make the algorithm faster. In our preliminary experiments, Sparse LPBoost solves ℓ_1 soft margin problems faster than the standard LP solver and LPBoost both in artificial and real data. Especially, for large datasets with ten thousands hypotheses and examples, Sparse LPBoost runs more than seven times faster than other algorithms.

There are some related researches. Warmuth et al. proposed Entropy Regularized LPBoost [16], a variant of LPBoost that approximately solves the soft margin optimization problem. Entropy Regularized LPBoost provably runs in $O(\log(m/\nu))/\varepsilon^2$ iterations, while a lowerbound of iterations of LPBoost is $\Omega(m)$ [15].

The algorithm proposed by Mangasarian [10] and the one proposed by Sra [13] add a quadratic term into the linear objective in the original LP problem and solve the modified quadratic program by Newton methods and Bregman's method (see, e.g., [3]), respectively. Their methods, unlike ours, does not take advantage of the sparsity of the underlying problem.

Bradley and Mangasarian [2] also proposed an algorithm that decomposes the underlying linear program into smaller ones, which seems similar to our idea. However, this algorithm only generates columns (hypotheses) as done in LPBoost.

2 Preliminaries

Let $\mathcal{X}$ be the domain of interest. Let $S = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m))$ be the given set of m examples, where each $\boldsymbol{x}_i$ is in $\mathcal{X}$ and each y_i is -1 or $+1$ $(i = 1, \ldots, m)$. Let $\mathcal{H}$ be the set of n hypotheses, where each hypothesis is a function from $\mathcal{X}$

to $[-1,+1]$. For any integer k, let $\mathcal{P}^k$ be the set of probability simplex, that is, $\mathcal{P}^k = \{\boldsymbol{p} \in [0,1]^k : \sum_{i=1}^k p_i = 1\}$. The margin of an example $(\boldsymbol{x}, y)$ w.r.t. a (normalized) hypothesis weighting $\boldsymbol{\alpha} \in \mathcal{P}^n$ is defined as $y_i \sum_{j=1}^n \alpha_j h_j(\boldsymbol{x}_i)$. Also, the margin of the set S of examples w.r.t. $\boldsymbol{w} \in \mathcal{P}^n$ is defined as the minimum margin of examples in S. The edge of a hypothesis $h \in \mathcal{H}$ w.r.t. a distribution $\boldsymbol{d} \in \mathcal{P}^m$ is defined as $\sum_{i=1}^m y_i d_i h(\boldsymbol{x}_i)$. For convenience, we denote $\gamma_{\boldsymbol{d}}(h)$ as the edge of a hypothesis h w.r.t. a distribution $\boldsymbol{d}$.

2.1 Linear Programming

A soft margin optimization problem with ℓ_1 regularization is formulated as follows (see, e.g., [5, 16]):

$$\max_{\rho, \boldsymbol{\alpha}, \boldsymbol{\xi}} \rho - \frac{1}{\nu} \sum_{i=1}^m \xi_i \quad (1)$$
$$\text{sub.to} \quad y_i \sum_j \alpha_j h_j(\boldsymbol{x}_i) \geq \rho - \xi_i \ (i = 1, \ldots, m),$$
$$\boldsymbol{\alpha} \in \mathcal{P}^n,$$

$$\min_{\gamma, \boldsymbol{d}} \gamma \quad (2)$$
$$\text{sub.to} \quad \sum_i d_i y_i h_j(\boldsymbol{x}_i) \leq \gamma \ (j = 1, \ldots, n),$$
$$\boldsymbol{d} \leq \frac{1}{\nu}\mathbf{1}, \boldsymbol{d} \in \mathcal{P}^m,$$

where the primal problem is given as (1) and the dual problem is given as (2), respectively. Let $(\rho^*, \boldsymbol{\alpha}^*, \boldsymbol{\xi})$ be an optimizer of the primal problem (1) and let $(\gamma^*, \boldsymbol{d}^*)$ be an optimizer of the dual problem (2). Then, by the duality of the linear program, $\rho^* - \frac{1}{\nu}\sum_{i=1}^m \xi_i^* = \gamma^*$. A notable property of the solution is its sparsity. By KKT conditions, an optimal solution satisfies the following property.

$$d_i^* \left(y_i \sum_j \alpha_j^* h_j(\boldsymbol{x}_i) - \rho^* + \xi_i^* \right) = 0 \quad (i = 1, \ldots, m)$$
$$d_i^* \geq 0, \ y_i \sum_j \alpha_j^* h_j(\boldsymbol{x}_i) - \rho^* + \xi_i^* \geq 0 \quad (i = 1, \ldots, m)$$
$$\xi_i^* \left(1/\nu - d_i^*\right) = 0, \quad \xi_i^* \geq 0, \ d_i^* \leq 1/\nu \quad (i = 1, \ldots, m)$$

This property implies that

- If $y_i \sum_j \alpha_j^* h_j(\boldsymbol{x}_i) > \rho^*$, then $d_i^* = 0$
- If $0 < d_i^* < 1/\nu$, then $y_i \sum_j \alpha_j^* h_j(\boldsymbol{x}_i) = \rho^*$.
- If $\xi_i^* > 0$, then $d_i^* = 1/\nu$.

Especially, an example $(\boldsymbol{x}_i, y_i)$ s.t. $d_i^* \neq 0$ is called a "support vector". Note that the number of inseparable examples (for which $\xi_i^* > 0$) is at most ν, since, otherwise, $\sum_i d_i^* > 1$.

Further, the primal solution has sparsity as well.

- If $\boldsymbol{d}^* \cdot \boldsymbol{u}_j < \gamma$, then $\alpha_j^* = 0$.

We call hypothesis h_j relevant if $\alpha_j^* > 0$. So, we can reconstruct an optimal solution by using only support vectors and relevant hypotheses.

Algorithm 1 LPBoost(S,ε)

```
1. Let $\boldsymbol{d}_1$ be the uniform distribution over $S$.
2. For $t = 1, \ldots,$
   (a) Choose a hypothesis $h_t$ whose edge w.r.t. $\boldsymbol{d}_t$ is more than $\gamma_t + \varepsilon$. Let $u_{t,i} = y_i h_t(\boldsymbol{x}_i)$.
   (b) If such a hypothesis doe not exist in $\mathcal{H}$, let $T = t - 1$ and break.
   (c) Solve the soft margin optimization problem (2) w.r.t. the restricted hypothesis set $\{h_1, \ldots, h_t\}$. Let $(\gamma_{t+1}, \boldsymbol{d}_{t+1})$ be the solution.
3. Output $f(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$, where each $\alpha_t$ $(t = 1, \ldots, T)$ is a Lagrange dual of the soft margin optimization problem (2).
```

3 Algorithms

In this section, we describe algorithms for solving the problem (2) in details.

3.1 LPBoost

First, we review LPBoost [5]. Given the initial distribution which is uniform over examples, LPBoost works in iterations. At each iteration t, LPBoost choose a hypothesis h_t with edge larger than $\gamma_t + \varepsilon$ w.r.t. $\boldsymbol{d}_t$, and add a new constraint $\boldsymbol{d} \cdot \boldsymbol{u}_t$, where $u_{t,i} = y_i h_t(\boldsymbol{x}_i)$ for $i = 1, \ldots, m$ to the current optimization problem and solve the linear program and get $\boldsymbol{d}_{t+1}$ and γ_{t+1}. We summarize the description of LPBoost in Figure 1.

For completeness, we show a proof that LPBoost can approximately solve the optimization problem (2).

Theorem 1 *LPBoost outputs a hypothesis whose soft margin is at least $\gamma^* - \varepsilon$.*

Proof. By definition of the algorithm, when LPBoost outputs the final hypothesis, it holds that $\gamma_T \geq max_{h \in \mathcal{H}} \gamma_{\boldsymbol{d}_T}(h) - \varepsilon$. Further, since $\boldsymbol{d}_t$ is a feasible solution of the dual problem (2), we have $max_{h \in \mathcal{H}} \gamma_{\boldsymbol{d}_T}(h) \geq \gamma^*$. Combining these facts, we obtain $\gamma_T \geq \gamma^* - \varepsilon$.

3.2 Our algorithm

Now we describe our algorithm Sparse LPBoost. Sparse LPBoost is a modification of LPBoost. There are two main differences. Fist difference is that the support of the distribution does not cover the entire set of examples, but covers the examples which have low margin with respect to the current hypothesis weighting. The second difference is that Sparse LPBoostcan choose more than two hypotheses at each iteration. The details of Sparse LPBoost is shown in Figure 2.

Then we prove the correctness of Sparse LPBoost.

Theorem 2 *Sparse LPBoost outputs a hypothesis whose soft margin is at least $\gamma^* - \varepsilon$.*

Algorithm 2 Sparse LPBoost(S,ε)

```
1. (initialization) Pick up $\nu$ examples arbitrarily and put them into $S_1$. Let $\gamma_1 = 1$.
2. For $t = 1, \ldots,$
   (a) Choose a set $S'_t$ of examples with margin w.r.t. $f_t$ less than $\rho_t$.
   (b) If there exists no such $S'_t$, then let $T = t - 1$ and break.
   (c) Let $S_{t+1} = S_t \cup S'_t$.
   (d) For $t' = 1, \ldots,$
       i. Choose a set $H'_{t'}$ of hypotheses whose edge is larger than $\gamma_t + \varepsilon$. Let $H_t = H_t \cup H'_{t'}$.
       ii. If there exists no such $H'_{t'}$, then let $f_t = f_{t'}$, $\rho_t = \rho_{t'}$ and break.
       iii. Solve soft margin LP problem (2) with respect to $S_t$ and $H_t$.Let $f_{t'+1}(\boldsymbol{x}) = \sum_{h \in H_t} \alpha_h h(\boldsymbol{x})$, where each $\alpha_h$ is the Lagrange dual of the problem (2), and $\rho_{t'+1}$ be the solution of the primal problem (1).
3. Output $f_T(\boldsymbol{x}) = \sum_{h \in H_T} \alpha_h h(\boldsymbol{x})$.
```

Proof. Let $C = S - S_T$. Since there is no hypothesis whose edge is more than $\gamma_T + \varepsilon$, $\gamma_T \geq \max_{h \in \mathcal{H}} \gamma_{\boldsymbol{d}_T}(h)$. Further, since $\boldsymbol{d}_T$ is a feasible solution of the problem (2), $\max_{h \in \mathcal{H}} \gamma_{\boldsymbol{d}_T}(h) \geq \gamma^*$, which implies $\gamma_T \geq \gamma^* - \varepsilon$. Now, consider the distribution $\boldsymbol{d}'_T = (\boldsymbol{d}_T, 0, \ldots, 0) \in \mathcal{P}^{|S|}$, which puts zero weights on examples in C. Then, it is clear that $(\gamma_T, \boldsymbol{d}'_T)$ satisfies the KKT conditions w.r.t. S.

3.3 Heuristics for choosing hypotheses and examples

So far, we have not specified the way of choosing hypotheses or examples. In this subsection, we consider some heuristics.

Threshold: Choose a hypothesis with edge larger than $\gamma'_t + \varepsilon$ and an example with margin less than ρ_t.

Max/min-one: Choose a hypothesis with maximum edge and an example with minimum margin.

Max/min-exponential: Let $\widehat{\mathcal{H}}_{t'}$ be the set of hypotheses whose edges with respect to $\boldsymbol{d}_{t'}$ are more than $\gamma'_t + \varepsilon$, and let $\widehat{S}_t$ be the set of examples whose margin is less than ρ_t. Then, choose the top K hypotheses with highest edges among $\widehat{\mathcal{H}}_{t'}$ and the top L examples with lowest edges among $\widehat{S}_t$, where K is $\min\{|\widehat{\mathcal{H}}_{t'}|, 2^{t'}\}$ and L is $\min\{|\widehat{S}_t|, 2^t\}$.

Let us consider which strategies we should employ. Suppose that $\nu = 0.2m$ and we use a linear programming solver which takes time is m^k, where k is a constant. Note that the value of ν is a reasonable choice since we allow at most 20% of examples to be misclassified.

If we take Threshold or Max/min-one approach, the computation time of Sparse LPBoost needs at least $\sum_{t=1}^{\nu} t^k > \int_{t=1}^{\nu} t^k d_t = \frac{\nu^{k+1}-1}{k+1} = \Omega(m^{k+1})$.

On the other hand, suppose we choose Max/min-exponential approach and the algorithm terminates when the number of chosen examples is cm $(0 < c < 1)$.

Then, the computation time is at most

$$\sum_{t=1}^{\lceil \log(cm) \rceil} (\nu + 2^t)^k = \sum_{s=0}^{k} \binom{k}{s} \nu^s \sum_{t=1}^{\lceil \log(cm) \rceil} 2^{t(k-s)} \leq \sum_{s=0}^{k} \binom{k}{s} \nu^s (c'm)^{k-s} = O(m^k).$$

Similar arguments hold for choosing hypotheses as well.

Therefore, we conclude that, among these approaches, Max/min-exponential approach is a more robust choice. Note that, the advantage of Sparse LPBoost is its small constant factor, e.g., $(0.2 + c')^k$. Even if the improvement is only by a constant factor, it might still influence the performance significantly.

4 Experiments

We compare LP, LPBoost and Sparse LPBoost for artificial and real datasets. Our experiments are performed on a workstation with a 8Gb RAM and Xeon 3.8GHz processors. We implemented our experiments with Matlab and use CPLEX 11.0, a state-of-the art LP solver.

Our artificial datasets contain from $m = 10^3$ to 10^6 instances in $\{-1, +1\}^n$. We fix a linear threshold function $f(\boldsymbol{x}) = x_1 + x_2 + \cdots + x_k + b$, which assigns a label($-1$ or $+1$) of each instance. We set $n = 100$, $k = 10$ and $b = 5$. For each data set, we generate instances randomly so that positive and negative instances are equally likely. Then we add 0% or 5 % random noise on labels.

For each dataset, we prepare $n + 1$ weak hypotheses. First n hypotheses correspond to the n th dimensions, that is $h_j(\boldsymbol{x}) = x_j$ for $j = 1, \ldots, n$. The last hypothesis corresponds to the constant hypothesis which always answers $+1$. We set $\nu = 1$ and $\nu = 0.2m$ for noise-free datasets and noisy datasets, respectively. For LPBoost and Sparse LPBoost, we set $\varepsilon = 0.01$.

We summarize the results for noise-free data and noisy data in Table 1 and **??**, respectively. Sparse LPBoost tend to run faster than others while approximating the solutions well. Note that, compared to other algorithms, the result of Sparse LPBoost is more robust with respect to the choice of ν. In addition, one can observe that, for both noise-free or noisy datasets, Sparse LPBoost picks up fewer examples than the total size m. As a result, the number of variables in the underlying problem is reduced, which makes computation faster. Also, as can be seen, Sparse LPBoost has fewer non-zero d_is when setting $\nu = 1$. On the other hand, Sparse LPBoost's computation time tends to increase when $\nu = 0.2m$. This is because the optimal distribution needs at least ν non-zero components.

Then we show experimental results for some real datasets. As real datasets, we use Reuters-21578 [1] and RCV1 [9].

For Reuters-21578, we use the modified Apte("ModApte") split which contains 10170 news documents labeled with topics. We create a binary classification problem by choosing a major topic "acq" as positive and regarding other topics as negative. As hypotheses, we prepare about $30,839$ decision stumps corresponding to words. That is, each decision stumps answers $+1$ if a given text contains the associated word and answers 0, otherwise.

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578.

$m = 10^3$	time(sec.)	$\#(d_i > 0)$	$\#(w_j > 0)$	time(sec.)	$\#(d_i > 0)$	$\#(w_j > 0)$
LP	**0.98**	96	96	**0.46**	217	46
LPBoost	5.46	83	83(84)	6.95	237	65(66)
Sparse LPBoost	2.38	26(520)	26(98)	4.80	243(655)	79(82)
$m = 10^4$	time(sec.)	$\#(d_i > 0)$	$\#(w_j > 0)$			
LP	29.66	101	101	**7.01**	2035	70
LPBoost	267.85	67	67(67)	21.51	2012	29(29)
Sparse LPBoost	**6.78**	25(5250)	25(98)	65.76	2031(6551)	58(58)
$m = 10^5$	time(sec.)	$\#(d_i > 0)$	$\#(w_j > 0)$			
LP	132.99	101	101	321.54	20046	92
LPBoost	1843.1	97	97(99)	71.65	200007	11(11)
Sparse LPBoost	**62.89**	22(50515)	21(94)	**60.51**	20006(64810)	11(11)
$m = 10^6$	time(sec.)	$\#(d_i > 0)$	$\#(w_j > 0)$			
LP	2139.3	101	101	39923	200031	60
LPBoost	17435	97	97(97)	1179	2000004	11(11)
Sparse LPBoost	**632.29**	22(439991)	22(100)	**1281.1**	200004(648771)	11(11)

Table 1. Summary of results for noise-free artificial data with $n = 100$, $\nu = 1$. For LPBoost and Sparse LPBoost, the numbers of chosen hypotheses or examples are shown in parentheses. LP, LPBoost and Sparse LPBoost obtained the same objective values γ.

Reuters-21578 (m=10,170,n=30,839)	time(sec.)	ρ ($\times 10^{-3}$)	γ ($\times 10^{-3}$)	$\#(d_i > 0)$	$\#(w_j > 0)$
LP	381.18	4.8	0.633	2261	463
LPBoost	804.39	4.8	0.633	2158	452(528)
Sparse LPBoost	**52.16**	4.8	0.633	2262(6578)	458(613)
RCV1 (m=20,242,n=47,237)	time(sec.)	ρ ($\times 10^{-3}$)	γ ($\times 10^{-3}$)	$\#(d_i > 0)$	$\#(w_j > 0)$
LP	2298.1	1.9	0.267	8389	639
LPBoost	2688.1	1.9	0.261	8333	454(465)
Sparse LPBoost	**235.63**	1.9	0.262	8335(16445)	480(518)

Table 2. Summary of results for real datasets. For LPBoost and Sparse LPBoost, the numbers of chosen hypotheses or examples are shown in parentheses.

For RCV1 data, we use the data provided by LIBSVM tools [4]. In the data, we consider binary classification problem by regarding the labels CCAT and ECAT as positive and labels GCAT and MCAT as negative. Each hypothesis is associated with a word and there are 47236 hypotheses in total. The output of hypothesis is given by the tf-idf weighting.

For both datasets, we add the constant hypothesis -1. We set $\varepsilon = 10^{-4}$ as the precision parameter of LPBoost and Sparse LPBoost. We specify $\nu = 0.2m$ and $\nu = 0.4m$ for Reuters-21578 and RCV1, respectively.

The results are summarized in Table 2. Sparse LPBoost runs several times faster than other algorithms. Like previous results for artificial datasets, Sparse LPBoost uses fewer examples (as many as about $0.6m$ to $0.8m$). Further, Sparse LPBoost seems to take advantage of the sparsity of relevant hypotheses as well. In both datasets, Sparse LPBoost chooses only about 600 hypotheses among more than $30,000$ hypotheses.

5 Conclusion

In this paper, we proposed a decomposition algorithm that approximately solves ℓ_1-soft margin optimization problems. Our algorithm performs faster than the standard LP solver using CPLEX and LPBoost by exploiting the sparsity of the underlying solution with respect to hypotheses and examples.

One of our future work is to modify Sparse LPBoost so as to have a theoretical guarantee of iteration bounds. Also, as a practical viewpoint, better heuristics for choosing hypotheses and examples should be investigated.

Acknowledgments

This research was supported by MEXT Grand-in-Aid for Young Scientists (B) 21700171.

References

1. N. Balcan, A. Blum, and N. Srebro. A theory of learning with similarity functions. *Machine Learning*, 72(1-2):89–112, 2008.
2. P. S. Bradley and O. L.Mangasarian. Massive data discrimination via linear support vector machines. *Optimization Methods and Software*, 13(1):1–10, 2000.
3. Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1998.
4. C. C. Chang and C. J. Lin. Libsvm: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.
5. A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.
6. T. Graepel, R. Herbrich, B. Schölkopf, A. Smola, P. Bartlett, K. Müller, K. Obermayer, and R. Williamson. Classification on proximity data with LP-machines. In *International Conference on Artificial Neural Networks*, pages 304–309, 1999.
7. A. J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *AAAI-98*, pages 692–698, 1998.
8. M. Hein, O. Bousquet, and B. Schölkopf. Maximal margin classification for metric spaces. *JCSS*, 71:333–359, 2005.
9. D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.
10. O. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *JMLR*, 7:1517–1530, 2006.
11. S. Nash and A. Sofer. *Linear and Nonlinear Programming*. Macgraw-Hill, 1996.
12. R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.
13. S. Sra. Efficient large scale linear progamming support vector machines. In *Machine Learning: ECML 2006*, pages 767–774, 2006.
14. L. Wang, M. Sugiyama, C. Yang, K. Hatano, and J. Fung. Theory and algorithms for learning with dissimilarity functions. *Neural Computation*, 2009, to appear.
15. M. Warmuth, K. Glocer, and G. Rätsch. Boosting algorithms for maximizing the soft margin. In *NIPS 20*, pages 1585–1592, 2008.
16. M. Warmuth, K. Glocer, and S. V. N. Vishwanathan. Entropy regularized LPBoost. In *ALT 2008*, pages 256–271, 2008.