Solving Semi-Infinite Linear Programs using Boosting-like Methods

Gunnar Rätsch

Friedrich Miescher Laboratory, Max Planck Society, Spemannstr. 39, 72076 Tübingen
Gunnar.Raetsch@tuebingen.mpg.de, http://www.fml.mpg.de/~raetsch

Linear optimization problems (LPs) with a very large or even infinite number of constraints frequently appear in many forms in machine learning. A linear program with m constraints can be written as

$$\begin{aligned} &\min_{\mathbf{x}\in\mathcal{P}^n} \quad \mathbf{c}^\top \mathbf{x} \\ &\text{with} \quad \mathbf{a}_j^\top \mathbf{x} \leq b_j \quad \forall i = 1, \ldots, m, \end{aligned}$$

where I assume for simplicity that the domain of $\mathbf{x}$ is the n dimensional probability simplex $\mathcal{P}^n$. Optimization problems with an infinite number of constraints of the form $\mathbf{a}_j^\top \mathbf{x} \leq b_j$, for all $j \in J$, are called *semi-infinite*, when the index set J has infinitely many elements, e.g. $J = \mathbb{R}$. In the finite case the constraints can be described by a matrix with m rows and n columns that can be used to directly solve the LP. In semi-infinite linear programs (SILPs) the constraints are often given in a functional form depending on j or implicitly defined, for instance by the outcome of another algorithm.

In this work I consider several examples from machine learning where large LPs need to be solved. An important case is *boosting* – a method for combining classifiers in order to improve the accuracy (see [1] and references therein). The most well-known instance is AdaBoost [2]. Under certain assumptions it finds a separating hyperplane in an infinite dimensional feature space with a *large margin*, which amounts to solving a semi-infinite linear program. The algorithms that I will discuss to solve the SILPs have their roots in the AdaBoost algorithm. The second problem is the one of learning to predict structured outputs, which can be understood as a multi-class classification problem with a large number of classes. Here, every class and example generate a constraint leading to a huge optimization problem [3]. Such problems appear for instance in natural language processing, speech recognition as well as gene structure prediction [4]. Finally, I consider the case of learning the optimal convex combination of kernels for support vector machines [5,6]. I show that it can be reduced to a semi-infinite linear program [7] that is equivalent to a semi-definite programming formulation proposed in [8].

I will review several methods to solve such optimization problems, while mainly focusing on three algorithms related to boosting: *LPBoost*, *AdaBoost** and *TotalBoost*. They work by iteratively selecting violated constraints while refining the solution of the SILP. The selection of violated constraints is done in a problem dependent manner: a so-called *base learning algorithm* is employed in boosting, dynamic programming is applied for structured output learning

and a single kernel support vector machine is used for multiple kernel learning. The main difference between optimization strategies is how they determine intermediate solutions. The first and conceptually simplest algorithm is *LPBoost* [9] and works by iteratively adding violated constraints to a restricted LP. The algorithm is known to converge [10,11,12] under mild assumptions but no convergence rates could be proven. The second algorithm, *AdaBoost** [13], is closely related to AdaBoost and works by multiplicatively updating the iterate based on the violated constraint. It was shown that this algorithm solves the problem with accuracy ϵ in at most $\lceil 2\log(n)/\epsilon^2 \rceil$ iterations. However, it turns out that *LPBoost*, which does not come with an iteration bound, is considerably faster than *AdaBoost** in practice. We have therefore worked on a new algorithm, called *TotalBoost* [14], that combines the advantages of both strategies: empirically it is at least as fast as *LPBoost* and it comes with the same convergence rates as AdaBoost*.

References

1. R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*, LNCS, pages 119–184. Springer, 2003.
2. Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
3. Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *Proc. ICML'03*, pages 3–10. AAAI Press, 2003.
4. G. Rätsch, S. Sonnenburg, J. Srinivasan, H. Witte, K.-R. Müller, R. Sommer, and B. Schölkopf. Improving the C. elegans genome annotation using machine learning. *PLoS Computational Biology*, 2006. Under revision.
5. C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
6. G. Lanckriet, N. Cristianini, L. Ghaoui, P. Bartlett, and M. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.
7. S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, pages 1531–1565, July 2006.
8. F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In C. E. Brodley, editor, *Proc. ICML'04*. ACM, 2004.
9. A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.
10. R. Hettich and K.O. Kortanek. Semi-infinite programming: Theory, methods and applications. *SIAM Review*, 3:380–429, September 1993.
11. G. Rätsch, A. Demiriz, and K. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1-3):193–221, 2002.
12. G. Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Neues Palais 10, 14469 Potsdam, Germany, October 2001.
13. G. Rätsch and M.K. Warmuth. Efficient margin maximization with boosting. *Journal of Machine Learning Research*, 6:2131–2152, 2005.
14. M.K. Warmuth, J. Liao, and G. Rätsch. Totally corrective boosting algorithms that maximize the margin. In W. Cohen and A. Moore, editors, *Proc. ICML'06*, pages 1001–1008. ACM Press, 2006.