CS 395T Computational Complexity of Machine Learning

Lecture 7: February 10, 2005 Lecturer: Adam Klivans

Scribe: Arvind Narayanan

7.1 Lower bounds on PTF degree for the intersection of two halfspaces

We have seen that: if the sum of weights is $\leq W$ then $O(\log W)$ is an upper bound on PTF degree for the intersection of two halfspaces. Now we will look at a lower bound.

Consider $h_1 = MAJORITY(x_1, x_2, \ldots, x_n)$. This is a halfspace (because it is the same as $x_1 + x_2 + \ldots + x_n \geq \frac{n}{2}$). Let $h_2 = MAJORITY(y_1, y_2, \ldots, y_n)$.

Theorem 1 (O'Donnell and Servedio '03) *Let h_1 and h_2 be as defined above. Any polynomial threshold function computing $h_1 \wedge h_2$ has degree $\frac{\log n}{\log \log n}$.*

This was the first improvement in lower bound after Minski and Papert's '68 result of $\omega(1)$ (interestingly enough, for the same function) [1].

7.2 Learning with a margin constraint

For bounded weight halfspaces, there are 3 ways of improving on the $O(n^{\log W})$ upper bound:

1. Restrict the sequence of samples in some way (such as force them to be drawn from the uniform distribution). This will be the second part of the course.
2. Allow the learner to have query access - i.e, it can choose examples at which to evaluate the unknown function
3. Apply a margin constraint. More on this now.

A margin is intuitively a measure of the distance between the collections of positive and negative examples. Suppose ρ is the margin. If it is a sufficiently large constant, then we have a poly time algorithm.

Let C be a concept class. (In this case C is the intersection of two halfspaces.)

Definition 1 (margin) *The margin of a set of points X each of which is in R^n is defined as:* $\rho = \frac{min_z \sum ||z-y||/z\in X, y\in R^n, C(z)\neq C(y)}{||\mathcal{X}||_\infty}$.

Note that $||\mathcal{X}||_\infty$ is defined as $max_x||x||_2$. Also note that y varies over all points, not just over $\mathcal{X}$. If we let y vary over $\mathcal{X}$ instead, we would get roughly twice the margin.

Usually we consider $\mathcal{X} \in \{0,1\}^n$.

Now we normalize the points and say that all the points are on S^{n-1}, the unit ball.

We also normalize the halfspaces and say that $||h_1|| = ||h_2|| = 1$. This does not change the halfspaces themselves but merely scales the coefficients. This simplifies the expression for the margin:

margin of h_1 wrt. $\mathcal{X} = min_{x\in\mathcal{X}} h_1(x)$.

We have seen that any set of integer weights can be replaced by another set whose sum is no more than $n^{\log n}$. An immediate consequence is that the margin can be no worse than $2^{-n\log n}$.

"Good" margins correspond to $\rho \approx 1$. We always have $0 < \rho \leq 1$.

ρ can be equivalently expressed in different ways, when $||x|| = 1$ and $||h|| = 1$:

- $\rho = min_{x\in\mathcal{X}} h(x)$
- $\rho = min_{\mathcal{X}} < h, x >$
- $\rho = min_{\mathcal{X}}$ distance from x from h.

7.3 Brute force algorithm for learning intersections of halfspaces

Let's assume we're in k dimensions and that there are m points. There is an $O(m^k)$ time brute force algorithm for learning the intersection of two halfspaces.

For each k-subset of points (of which there are $O(m^k)$), check if the halfspace passing through those points classifies all the positive examples as positive (it doesn't matter if some negative examples are incorrectly classified). Each such halfspace is a candidate. After finding all candidates, consider all pairs of candidates and the corresponding intersections. At least one these will classify all the examples correctly, and it is our answer.

Why does this algorithm work? Because a halfspace can be "pushed" towards the positive examples until it touches k of them, since we are in a k-dimensional space.

Summarizing,

Theorem 2 *There exists an algorithm with a running time of $O(m^k)$ which outputs an intersection of two halfspaces consistent with any labeling of m points in $\mathbb{R}^k$, provided that one exists.*

How do we generalize this to intersections of t halfspaces? In the obvious manner: we obtain the candidates as above and consider all t-subsets of candidates. This gives us a $O(m^{kt})$ algorithm.

Theorem 3 *There exists an algorithm with a running time of $O(m^{kt})$ which outputs an intersection of t halfspaces consistent with any labeling of m points in $\mathbb{R}^k$, provided that one exists.*

References

1. M. Minsky and S. Papert. *Perceptrons.* MIT Press 1968.
2. R. O'Donnell and R. Servedio. *Learning intersections and thresholds of halfspaces.* . In Proceedings of the Forty-Third Annual Symposium on Foundations of Computer Science, pages 177-186, 2002.
3. R. Beigel, N. Reingold, D. Spielman. *PP is closed under intersection.* Proceedings of the twenty-third annual ACM symposium on Theory of computing, pages 1-9, 1991.