CS 395T Computational Learning Theory

Lecture 1: September 7, 2005 — Lecturer: Adam Klivans

Scribe: Lily Mihalkova

1.1 Mistake-Bounded Model; Simplified Winnow; Best Experts

1.1.1 Mistake-Bounded Model

In the Mistake-Bounded Model, the goal is to learn an unknown function from a given concept class. For example, one function from the class of logical disjunctions could be $x_1 \vee x_3 \vee \overline{x_5}$. The learner sees a sequence of labeled examples. After seeing example i, which comes in the format $(example_i, label_i)$, the learner outputs a hypothesis, which labels the example positive or negative. If the hypothesis mislabels the example, the learner made a mistake. In this case, it updates its hypothesis. The next example is then provided.

The learner has a mistake bound of t if for all (possibly infinite) sequences of example/label pairs, it makes at most t mistakes. The learning algorithm is considered efficient if it runs in time polynomial in the length of the examples and the unknown concept.

1.1.2 Learning Disjunctions

As we saw last time, learning disjunctions on n variables has a mistake bound of $O(n)$. The algorithm proceeds as follows. The initial hypothesis is the most general disjunction that consists of all variables and their negations:

$$x_1 \vee \overline{x_1} \vee x_2 \vee \overline{x_2} ... \vee \overline{x_n}$$

Because we start with the most general hypothesis and assume that there is no noise in the examples, the only type of mistake the algorithm can make is where it labels a negative example positive. When this happens, the hypothesis is updated by removing every non-negated variable that was set to 1 in the example and every negated variable that was set to 0 because they cannot possibly be in the disjunction (otherwise, the label would have been positive and a mistake would not have been made). Because the original disjunction had length $2n$ and at each mistake we remove at least one variable from it, the mistake bound of this algorithm is $2n$.

Now consider the scenario where we know that the disjunction is on only $k << n$ variables. In this case, the mistake bound can be improved to $O(k \log n)$ using the following algorithm, which is a simplification of the Winnow algorithm proposed by Nick Littlestone.

Assume for simplicity that the unknown disjunction contains no negations. The initial hypothesis is

$$x_1 + x_2 + \dots x_n \geq n$$

Update 1: Suppose that the algorithm labels a positive example negative. In this case we double the coefficient of each variable with a 1 in its position.

Update 2: If on the other hand the algorithm labels a negative example positive, we cut in half the coefficient of each variable with a 1 in its position.

To determine the mistake bound of this algorithm, we make use of the following observation. The number of times each correct variable is doubled in update 1 is $\log n$. After this, it reaches n and dominates the expression, thus ensuring that a positive example is labeled correctly. There are at most k variables participating in the unknown disjunction. Thus, at most $k \log n$ updates of type 1 are made.

Now let us consider the number of updates of the second type that are made. Let W denote the sum of all the coefficients. Initially, $W = n$. For update 1, W is increased by an amount that is $\leq n$. For update 2, W is decreased by at least $n/2$. Thus, the number of decreases is $\leq$ twice the number of increases. We already determined that the number of increases is $\leq k \log n$. Thus, the number of type 2 updates is $\leq 2k \log n$.

Therefore, the total mistake bound is $3k \log n$.

Note: For the case of learning disjunctions, update 2 can be modified so that the coefficient of each variable with a 1 in its position is set to 0.

1.1.3 Best Experts

In this setting, we have k experts $C_1, \ldots, C_k$. As before, the learner sees a sequence of examples. Each expert is allowed to learn independently from the examples. The hypothesis will be formed as a linear theshold function of the outputs of the experts:

$$w_1 C_1(x) + w_2 C_2(x) + \ldots + w_k C_k(x) \geq 0$$

The above assumes that $C_i(x)$ is either 1 or -1.

The Weighted Majority algorithm gives a mistake bound that is close to that of the best expert. It works as follows. Initially, all the weights are set to 1. Prediction is based on the weighted average of the experts as given by the inequality above, and at each step the coefficient of each expert that makes a mistake is cut in half.

Let us now analyze the mistake bound of this algorithm. Let the total weight W be defined as the sum of all the weights. Initially, $W = k$. If a mistake is made in the first step, at least $k/2$ of the experts are wrong. After their weight is cut in half, the remaining weight is $W \leq 3/4k$. After M mistakes, the total weight will be $W \leq k(3/4)^M$ (because after a mistake is made at most 3/4 of the original weight is left).

Now let m be the number of mistakes made by the best expert. The weight of this expert is then

$(1/2)^m$. This weight has to be $\leq W$. Thus,

$$\begin{aligned}
\left(\tfrac{1}{2}\right)^m &\leq k\left(\tfrac{3}{4}\right)^M \\
\left(\tfrac{4}{3}\right)^M &\leq k \cdot 2^m \\
M &= O(m + \log k)
\end{aligned}$$

In other words, the mistake bound of the weighted majority algorithm is $O(opt + \log k)$ where opt is the mistake bound of the best expert. This algorithm is especially useful in a case when we have several predictors but are not sure which one is the best.

It is interesting to consider what happens if each expert is one of the concepts in the concept class. In this case the best expert is the unknown concept and has mistake bound of 0. Thus the mistake bound of the algorithm will be logarithmic in the size of the concept class.

There is also a randomized version of the algorithm with expected mistake bound of $(1+\epsilon)opt + 1/\epsilon \log k$.