8803 Machine Learning Theory

Maria-Florina Balcan Lecture 4: September 1st, 2011

Plan: Discuss the Mistake Bound model.

The Mistake Bound model

In this lecture we study the *online learning* protocol. In this setting, the following scenario is repeated indefinitely:

1. The algorithm receives an unlabeled example.
2. The algorithm predicts a classification of this example.
3. The algorithm is then told the correct answer.

We will call whatever is used to perform step (2), the algorithm's "current hypothesis."

Definition 1 *An algorithm A is said to* learn C in the mistake bound model *if for any concept $c \in C$, and for any ordering of examples consistent with c, the total number of mistakes ever made by A is bounded by $p(n, size(c))$, where p is a polynomial. We say that A is a* polynomial time *learning algorithm if its running time per stage is also polynomial in n and $size(c)$.*

Let us now examine a few problems that are learnable in the mistake bound model.

Conjunctions

Let us assume that we know that the target concept c will be a conjunction of a set of (possibly negated) variables, with an example space of n-bit strings.

Consider the following algorithm:

1. Initialize hypothesis h to be $x_1\overline{x_1}x_2\overline{x_2}\ldots x_n\overline{x_n}$.
2. Predict using $h(x)$.
3. If the prediction is **False** but the label is actually **True**, remove all the literals in h which are **False** in x. (So if the first mistake is on 1001, the new h will be $x_1\overline{x_2}\,\overline{x_3}\,x_4$.)

4. If the prediction is **True** but the label is actually **False**, then output "no consistent conjunction".

5. Return to step 2.

An invariant of this algorithm is that the set of literals in c will always be a subset of the set of literals in h.

The first mistake on a positive example will bring the size of h to n. Each subsequent such mistake will remove at least one literal from h, so that the maximum number of mistakes made will be at most $n + 1$.

Lower Bound: In fact no deterministic algorithm can achieve a mistake bound better than n in the worst case. This can be seen by considering the sequence of n examples in which the ith example has all bits except the ith bit set to 1. The target concept c will be a monotone conjunction constructed by including x_i only if the algorithm predicts the ith example to be **True** (in which case the ith example's label will be **False**). (If the algorithm predicts the ith example to be **False**, then the target concept will not include x_i, and so the true label will be **True**.) The algorithm will have made n mistakes by the time all of these n examples are processed.

Decision lists

As discussed in a previous lecture a *Decision List* is a list of if-then rules where each condition is a literal (a variable or its negation). Formally,

Definition 2 *A* **decision list** *is a list of rules:*

$$A_1 \to B_1,\ A_2 \to B_2,\ \ldots,\ A_{l-1} \to B_{l-1},\ True \to B_l$$

where A_i are literals and $B_i \in \{0, 1\}$. The meaning is: given an example, we look at the first left-hand-side satisfied and output the corresponding right-hand-side.

We present here an algorithm for learning in the mistake bound model. Note that in this algorithm our hypotheses will *not* belong to C.

In particular, our hypothesis will be a list of *subsets* of $\{x_1 \Rightarrow$ **True**, $x_1 \Rightarrow$ **False**, $\overline{x_1} \Rightarrow$ **True**, $\overline{x_1} \Rightarrow$ **False**, $x_2 \Rightarrow$ **True**, $\ldots$ $\overline{x_n} \Rightarrow$ **False**, $T \Rightarrow$ **True**, $T \Rightarrow$ **False**$\}$. The algorithm is the following.

1. Let h be the one-level list, whose level consists of the single set that includes all $4n+2$ possible terms.

2. Given an example x, look at the first set in h for which x satisfies the antecedents of at least one of the rules of the set. (I.e., select some rule that fires.) Predict based on this rule, breaking ties arbitrarily.

3. If the prediction is mistaken, move all the rules in that set which predicted incorrectly to the next set in the list.

4. Return to step 2.

Example: As an example of running this algorithm, say that n is 2 and the target concept is:

$$x_1 \Rightarrow \textbf{False}, \text{ else } \overline{x_2} \Rightarrow \textbf{True}, \text{ else } \textbf{False}.$$

Initially h will be

$$\{ \text{ all rules } \}.$$

If the first example is 01 then the algorithm will see two possible predictions in the first set of h; say it chooses to predict positively. In fact, though, this is a negative example. So h will be changed to

$$\{x_1 \Rightarrow \textbf{True}, x_1 \Rightarrow \textbf{False}, \overline{x_1} \Rightarrow \textbf{False}, x_2 \Rightarrow \textbf{False}, \overline{x_2} \Rightarrow \textbf{True}, \overline{x_2} \Rightarrow \textbf{False}, T \Rightarrow \textbf{False}\},$$

$$\text{else } \{\overline{x_1} \Rightarrow \textbf{True}, x_2 \Rightarrow \textbf{True}, T \Rightarrow \textbf{True}\}.$$

Now say that we give an example of 00 to the algorithm. Then the algorithm might choose to predict that this will be negative, though in fact it is positive. So h will now be

$$\{x_1 \Rightarrow \textbf{True}, x_1 \Rightarrow \textbf{False}, x_2 \Rightarrow \textbf{False}, \overline{x_2} \Rightarrow \textbf{True}\},$$
$$\text{else } \{\overline{x_1} \Rightarrow \textbf{True}, \overline{x_1} \Rightarrow \textbf{False}, x_2 \Rightarrow \textbf{True}, \overline{x_2} \Rightarrow \textbf{False}, T \Rightarrow \textbf{True}, T \Rightarrow \textbf{False}\}.$$

And so on.

Mistake Bound: We prove a mistake bound of $O(nL)$ by making several observations about this algorithm. First, with each mistake at least one term will move one level lower in h. Second, the ith rule of c will never move below the ith level of h – this can be proven by induction. And finally, each rule will fall at most $L + 1$ levels (where L is the length of target c). These three properties imply that with every mistake the algorithm always makes progress toward a correct solution. Furthermore, the number of mistaken predictions is bounded above by $(4n + 2)(L + 1)$. This is polynomial in n and the size of c, so this algorithm learns decision lists in the mistake bound model.