8803 Machine Learning Theory

Maria-Florina Balcan — Lecture 6: September 8, 2011

1 Relating the Mistake Bound and PAC models

We can relate the mistake bound and PAC models as follows:

Theorem 1 *If algorithm $\mathcal{A}$ learns a concept class C in the mistake bound model, then $\mathcal{A}$ also learns C in the probably approximately correct model.*

Proof: In proving this we would like to assume that an algorithm is *conservative* (or lazy), meaning that the algorithm only changes its state (its state is its current hypothesis) when it makes a mistake. We can insure that it does this by taking a snapshot of its state before giving it an example; if the algorithm predicts correctly, we will restore the state it had before the prediction. In this case the algorithm will behave as it would were it only given examples on which it makes mistakes, so the number of mistakes is still bounded. Thus, given $\mathcal{A}$ learning C in the mistake bound model we can construct $\mathcal{A}'$ also learning C in the mistake bound model but only changing its hypothesis when it makes a mistake.

Transforming the algorithm to a lazy one is not strictly necessary, but it makes the proof simpler.

From $\mathcal{A}'$ we can construct the following algorithm $\mathcal{A}_{PAC}$ (where M is the mistake bound, which is bounded by a polynomial in n and the size of c):

> Run $\mathcal{A}'$ on the data seen, halting if any hypothesis h survives for more than $\frac{1}{\epsilon}\ln(\frac{M}{\delta})$ subsequent examples. Return h as the hypothesis of the algorithm.

Note that since the number of mistakes made is bounded by M, the algorithm will terminate within $\frac{M}{\epsilon}\ln(\frac{M}{\delta})$ examples.

The probability that the algorithm accepts a hypothesis with error greater than ϵ is at most

$$M(1-\epsilon)^{\frac{1}{\epsilon}\ln(\frac{M}{\delta})} < Me^{-\frac{\epsilon}{\epsilon}\ln(\frac{M}{\delta})} = M\frac{\delta}{M} = \delta$$

So the chance that an algorithm accepts a good hypothesis (of error at most ϵ) is at least $1-\delta$; hence $\mathcal{A}_{PAC}$ learns in the PAC model. ∎

2 The Halving Algorithm

If we do not care about computation time or the hypothesis space H, and if C is a finite set, then we can achieve a mistake bound of $\lg |C|$. This is achieved by the following method called the “halving algorithm”:

1. Initialize V to C. (V is called the version space.)
2. Given example x, predict according to the *majority* of the concepts in V.
3. Remove from V all the concepts in C that predicted wrongly.
4. Return to step 2.

Since with each mistake at least half of the version space V is removed, the number of mistakes is bounded by $\log |C|$.

Clearly this is a very nice general algorithm to learn an arbitrary concept space in both the mistake bound model and the PAC model. Unfortunately, though, storing V and predicting according to the majority of the concepts in V is likely to be hard.

3 Summary of Learning Models for the Realizable Case

So far we have focused on the *realizable case* – the algorithm gets a sample that is consistent with a function in a fixed concept class C. The table below summarizes some of the classes that are learnable or not learnable in the learning models we have discussed so far.

Table 1: Learnable/not learnable in *polynomial time*; k is assumed to be constant.

	Consistency model	Mistake Bound Model	PAC model
Disjunctions	Yes	Yes	Yes
Conjunctions	Yes	Yes	Yes
k-CNF	Yes	Yes	Yes
k-term DNF	No ($NP \neq RP$ assumption)	Yes	Yes
k-Decision Lists	Yes	Yes	Yes
Linear separators	Yes	Yes	Yes
Blum’91	Yes	No (crypto assumption)	Yes
poly size circuits	Yes	No (crypto assumption)	No (crypto assumption)
Decision Trees	Yes	Not known	Not known
DNF	Yes	Not known	Not known

We have also seen some general relationships between these models. For example, if C is learnable in the consistency model and $\ln(C)$ is $poly(n)$, then C is learnable in the PAC model. If C is learnable in the mistake bound model, then C is learnable in the PAC model.

If C is the class of linear separators of "large" L_2 margin (i.e., L_2 margin is at least $1/poly(n)$), then C is learnable in the mistake bound model via the Perceptron algorithm. If C is the class of linear separators of "large" L_1 margin, then C is learnable in the mistake bound model via the Winnow algorithm. The general class of linear separators is learnable via an ellipsoid style algorithm in the mistake bound model.

Under crypto assumptions, there exist classes that are learnable in the PAC model, but not in the mistake bound model. One example is the Blum'91 class. The rough idea of this construction is that the target c is negative except for a special subset $X_1 = \{x^1, x^2, ..., x^m\}$ of $0, 1^n$ with the following special property:

1. the example x^i contains inside it enough information to determine the value $c(x^j)$ for all $j > i$, but without giving away c(x^j) for any $j < i$.

2. it is easy to determine in poly time whether an example belongs to X_1 or not (and if so, which one it is in the ordering).

So, in the mistake bound model, if the adversary presents the examples in reverse order x^m, x^{m-1},..., then the poor algorithm can't figure anything out. But, in the PAC model, given any distribution, we can just look at the one of smallest index j in X^1 and this tells us what c is on all larger index examples (so the only region of uncertainty is $x^1, ..., x^{j-1}$).