CS 598: Theoretical Machine Learning

Lecturer: Pranjal Awasthi
Scribe: Neelesh Kumar

Lecture #8
10/3/2017

1 Motivation for Online Learning

In the case of supervised learning, we have access to a large amount of training data, and typically the algorithm has a lot of time to process the training data. The crucial underlying assumption is that there is a relationship between test data and training data. However there are scenarios in which the assumptions made by supervised learning can fail:

- Training data comes in online fashion (one by one), and the predictions need to be made in real time. Examples of this scenario include online advertising, recommendation engines, etc.
- Test data may not be drawn from the same distribution as training data. For example- one may teach a robot to navigate in a particular environment, but ask it to navigate in a different environment. Another example is machine learning in the presence of adversaries- classifying emails as spam. No matter how good the algorithm is, spammers can adapt to this algorithm and fool it.
- Yet another motivation is to build a non probabilistic theory of machine learning.

2 Setup: Mistake Bound Model

The objective is to approximate a target function $h : X \mapsto \{+1, -1\}$. At each time step:

1. Algorithm sees an example from instance space, $x_t \in X$.
2. Algorithm predicts the label $y_t \in \{+1, -1\}$.
3. Algorithm sees the true label.

The algorithm doesn't make any assumptions about the distribution of the sample. In PAC learning, the notion of the performance of the algorithm was $err(f) = Pr_{X \sim D}(f \neq h)$. For online learning, this definition does not make any sense since there is no distribution D involved. Instead, the performance measure in this case is the number of mistakes an algorithm makes on any input sequence. This is also known as the **mistake bound** of the algorithm. For an algorihm to have learnt something, the mistake bound must be finite if ran for infinite time.

3 Example: Boolean Disjunctions

We wish to design an algorithm for the boolean disjunctions problem over $\{0, 1\}^n$. Let $h : x_1 \vee x_2 \vee \overline{x_4}$ be the target function. An approach can be to start with boolean disjunctions over all variables:

$$f_0 = x_1 \vee x_2 \vee .... \vee x_n \vee \overline{x_1} \vee \overline{x_2} \vee ....\overline{x_n}$$

Suppose f_0 makes a mistake on the example $\vec{X}_t$, for which $h(\vec{X}_t) = -1$ but $f_0(\vec{X}_t) = +1$. We remove any literal that is set to 1 in $\vec{X}_t$. Hence

$$f_1 = f_0 | \{\text{any literal that is set to 1 in } \vec{X}_t\}$$

The algorithm maintains the property that the true function is always a part of f_t. So the algorithm never makes any mistake on positive examples. Hence mistake bound for the algorithm is $O(2n)$.

4 Formal Definition

Given $h \in H$, and algorithm A,

$mb_A(h)$= mistake bound of A on h

$mb_A(H)= max_{h \in H}\ mb_A(h)$. This is the mistake bound of learning on any given function of class H

$MB(H)= min_A\ mb_A(H)$. This is the mistake bound achieved by the best algorithm for the class of function H.

Theorem 1. *If $h \in H$ and H is finite, then mistake bound $MB(H) \leq \log_2(|H|)$.*

Proof : First note that in the case of finite H, a weaker bound on the mistake bound is $MB(H) \leq |H|$. We start with an arbitrary function in H. If it makes a mistake on any point, we throw it away and pick another function.

To prove the theorem, we present **Halving Algorithm**:
Given an example from the instance space at any time step, predict according to the majority vote of all functions in the function class H.

1. Start with $U = H$.
2. On receiving an input x_t: $U_1 = \{h \in U; h(x_t) = +1\}$ and $U_2 = \{h \in U; h(x_t) = -1\}$. If $|U_1| \geq |U_2|$, predict +1, else predict -1.
3. Throw away the functions that make a mistake on x_t.

After each mistake, $|U_{t+1}| \leq \frac{1}{2}|U_t|$. Also the true function is never thrown away. Hence the mistake bound for the algorithm is $MB(H) \leq \log_2(|H|)$.

Theorem 2. *For any H, $MB(H) \geq d$, where d is the VC dimension of H.*

Proof : If the VC dimension of H is d, then $\exists$ d points $x_1, x_2, ..., x_d$ that can be shattered. An adversary can always say that the algorithm is wrong in its prediction on these points, no matter what the algorithm predicts and commit to a different labeling than what the algorithm says. Since all possible labellings on the d points are realizable, the labeling that the adversary commits to will always be realizable.

5 Online learning vs. PAC learning

Online learning is **harder** than PAC learning. To prove this statement, the following statements need to be proved:

1. $\exists H$ such that H can be PAC learned but cannot be learned in online learning model, i.e. $MB(H) = \infty$.

2. If H can be learned in online model, then it can also be PAC learned.

To prove the first statement, it is sufficient to provide an example of a function that can be PAC learned but for which $MB(H) = \infty$. One such class of functions is the threshold function. For the function to be learned under mistake bound model, the algorithm needs to make finite number of mistakes in infinite time, after which it makes no mistakes. However this is not possible for threshold functions, since in order to make no mistakes, the algorithm will have to see all the points. In general, if the class of functions H is infinite, then it cannot be learned in online model.
Next we prove the second statement.

Theorem 3. *If $MB(H) = B$, then H can be PAC learned with $m \geq \frac{B}{\epsilon}\log(\frac{B}{\delta})$ where symbols have their usual meaning.*

Proof : The idea is to feed the training data to the online algorithm, one at a time. If a function does well for a long time, i.e. classifies a sequence of $\geq \frac{m}{B}$ examples correctly, then we output it as a good function, otherwise we switch to another function. This switching will happen at most B times, by definition of mistake bound. Let h_i be a function that satisfies the condition in the previous statement but for which $err(h_i) > \epsilon$ under PAC model.

$$Pr(h_i \text{ does well on } \geq \tfrac{m}{B} \text{ examples}) \leq (1-\epsilon)^{\frac{m}{B}} \text{ which is } \leq e^{\frac{-\epsilon m}{B}}$$

Because there are at most B such bad functions, using union bound we get:

$$Pr(\text{the process outputs a bad function}) \leq Be^{\frac{-\epsilon m}{B}}$$
$$\text{Set } Be^{\frac{-\epsilon m}{B}} \leq \delta$$
$$m \geq \frac{B}{\epsilon}\log(\frac{B}{\delta})$$

6 Online Learning when $h \notin H$

So far, we have assumed that the true function will never make an error. But it could be quite possible that there are not any function in the given function class H that has zero error. If we run the halving algorithm, it could be possible that we throw all the functions away. In such a case, we wish to provide guarantees on how the function returned by an online algorithm compares against the best function in the class.

Theorem 4. *On any request sequence (input), let m be the mistake bound of best function h in the class H. Let M be the mistake bound of the halving algorithm. We have,*

$$M \leq m * (\log(|H|)$$

The idea is to run the halving algorithm on the input sequence. If all the functions in the class are thrown away, then we restart the halving algorithm. In each run of the halving algorithm, we make $\leq \log(|H|)$ mistakes. Also in each run of the halving algorithm, the best function will make at least one mistake. Since the best function makes m mistakes, the halving algorithm will be run $\leq m$ times. Hence

$$M \leq m * (\log(|H|)$$

7 References

- http://www.cs.cmu.edu/ avrim/Papers/survey.pdf