CS787: Advanced Algorithms

Scribe: Archit Gupta, Pavan Kuppili **Lecturer:** Shuchi Chawla
Topic: Weighted Majority, Mistake Bound Model, and Winnow Algorithm **Date:** 11/12/2007

23.1 Prediction Problem

Last class we looked at the technique of maximizing reward using Multiplicative Update technique. Today we look at a similar problem for online learning, where our goal is to minimize costs.
Let there be n experts giving predictions whether it'll rain or not. Expert i predicts $p_i \in \{-1, 1\}$ where $p_i = 1$ if the prediction is it'll rain and vice versa. Each expert i is associated with a weight w_i.

Now we have to decide whether it'll rain. Earlier on, we picked an expert to bet on. Note that instead of picking an expert with probability proportional to w_i, here we take a weighted vote of experts. This algorithm is called the Weighted Majority algorithm.

Algorithm: Let $w_i = 1$ at time $t = 0$. After each instance 't' of a prediction, for each of the expert i, that make a mistake: $w_i \leftarrow w_i(1 - \epsilon)$.
To predict the next event $t + 1$, we take the weighted vote of the experts.
We pick value 1 if $\sum(w_i p_i) > 0$
We pick value -1 if $\sum(w_i p_i) < 0$,
where w_i is the weight associated with the expert i and p_i is the prediction made by the expert i.

23.1.1 General case

We can generalize the above problem by letting the costs take any continuous value between 0 and 1. Now, each expert i makes a prediction $p_{i,t}$ which can lead to a loss $l_{i,t} \in [0, 1]$ at time t of the prediction. Our goal remains to minimize our costs.

Let $w_{i,t}$ be the weight of expert i at time t, $l_{i,t}$ be the loss of expert i at time t, and let W_t be the total weight of all experts at time t. The algorithm is as follows.

Algorithm:
For each expert initialize $w_{i,0} = 1$.
At each step, pick expert i with prob. $\frac{w_{i,t}}{W_t}$, where $W_t = \sum_i(w_{i,t})$
Update $w_{i,t+1} = w_{i,t}(1 - \epsilon)^{l_{i,t}}$ where $l_{i,t}$ is the loss of expert i.

Claim 23.1.1 *$W_T \geq (1 - \epsilon)^L$, where T is the final time step, and L is the total loss of the best expert.*

Proof: Let n be the total number of experts, and let j be the best expert. So $W_T = \sum_{i=1}^{n} w_{i,T} \geq w_{j,T} = (1 - \epsilon)^L$. ■

Let F_t be the expected cost of the algorithm at time t. So $F_t = \sum_i \frac{w_{i,t} l_{i,t}}{W_t}$
Our total expected cost $= \sum_t \sum_i \frac{w_{i,t} l_{i,t}}{W_t}$
Now let us get a relation between F_t and W_T, so that using claim 1, we can get a relationship between F_t and L.

$$W_{t+1} = \sum_i w_{i,t}(1-\epsilon)^{l_{i,t}}$$

$$\leq \sum_i w_{i,t}(1-\epsilon l_{i,t})$$

$$= \sum_i w_{i,t} - \epsilon \sum_i w_{i,t} l_{i,t}$$

$$= W_t - \epsilon W_t F_t$$

$$W_{t+1} \leq W_t(1-\epsilon F_t)$$

This gives,

$$W_T \leq n \prod_t (1-\epsilon F_t)$$

Using claim 1,

$$(1-\epsilon)^L \leq n \prod_t (1-\epsilon F_t)$$

$$L \log(1-\epsilon) \leq \log n + \sum_t \log(1-\epsilon F_t)$$

(Note that $\log(1+x) \leq x$)

$$L \log(1-\epsilon) \leq \log n - \epsilon \sum_t F_t$$

$$\sum_t F_t \leq -L \log(1-\epsilon)/\epsilon + \log n/\epsilon$$

If ϵ is small, using Taylor's series expansion, we get our total expected cost $= \sum_t F_t \approx (1 + \epsilon/2)L + \log n/\epsilon$ If instead of loss we were to gain G, the equivalent equation becomes $\sum_t F_t \leq (1-\epsilon/2)G - 1/\epsilon \log n$
Now, let $\epsilon = \sqrt{\log n/L}$. If L is large enough, ϵ is small and we get,

$$\sum_t F_t \approx L + O(\sqrt{L \log n})$$

This is a good result as our expected cost is within an additive term of $O(\sqrt{L \log n})$ of the best expert.

23.2 Mistake Bound Model for Learning

This is a variant of the prediction problem in which we believe that atleast one expert is always correct. The goal is to minimize the number of mistakes our algorithm makes before we converge to the "correct" expert. Note that the above multiplicative update method gives us a bound of $1/\epsilon \log n$ in this setting as $L = 0$. Can we do better?

A simple naive algorithm would be to choose a random expert at each step, and eliminate all those experts who make a mistake. But in the worst case, this algorithm will end up making $(n-1)$ mistakes, where n is the number of experts. (The worst case is when in each step, only the expert we pick makes a mistake).

23.2.1 Halving algorithm

This algorithm makes a total of only $\log_2 n$ mistakes. The algorithm is as follows.
1. Start with all the experts.
2. For each example, take a majority vote of the remaining experts, and eliminate all the experts who make a mistake.
3. Keep repeating step 2 for each incoming example on which we need to make a prediction.

It is easy to note that whenever we make a mistake, we remove atleast half of the remaining experts (since we voted in accordance with the majority of these experts). So the halving algorithm makes atmost $log_2 n$ mistakes before we eliminate all but the "correct" expert, and from then on we never make a mistake.

23.3 Concept Learning

Often times, we can never maintain an explicit list of all the experts and do the Halving algorithm. So in this section, we will look at implicit ways of doing the same thing (without maintaining an explicit list of all remaining experts). For example, if we take the spam example, each email can have multiple attributes. For example, 'contains the word Viagra' can be one attribute, 'fraction of spelling mistakes > 0.2' can be one attribute, and so on. An expert is a function over these attributes. Concept class is the class of all the experts we consider. For example, one concept class can be the class of decision trees with at most n nodes. While this concept class is smaller than the class of all possible experts, it is still very huge (exponential) and we cannot use the explicit halving algorithm. Next we consider two algorithms on specific concept classes in the mistake bound model.

23.3.1 Concept class of OR functions

Here each expert is an OR of some of the attributes. So, if we have n attributes, there are a total of 2^n experts (each attribute may or may not be present in the expert). The mistake bound algorithm is as follows.
1. Start with a list L of all the attributes.
2. We predict accoring to the OR of all the remaining attributes in L.
3. If we make a mistake on an example e, we remove all the attribute in e from our list L.

4. Keep repeating steps 2 and 3 for each example on which we need to make predictions.

Theorem 23.3.1 *The number of mistakes made by the above mistake bound algorithm on the concpt class of OR functions $\leq n$, where n is the number of attributes.*

Proof: It can be noted that we maintain the invariant that L is a superset of all the attributes in the true expert. If an example contains atleast one attribute in the true expert, the example is true, and we also predict true. We never end up removing an attribute in the true expert. However an example might be false, and we might predict it as true. (For example in the beginning, when we have all the attributes in L). However, each time we make a mistake, we remove atleast one attribute. So we make atmost n mistakes before we elminate all the attributes not in the true expert. ■

23.3.2 Concept class of OR functions of maximum size k

Here, $|C| = \sum_{i=0}^{k} \binom{n}{k} \approx n^k$. So, we want a mistake bound model which has a maximum of $O(k \log n)$ mistakes. Let A be the set of n attributes. Let w_i be the weight of the i^{th} attribute which we keep updating. In any example, let $x_i = 1$, if the example contains the i^{th} attribute, and $x_i = 0$, otherwise. The following Winnow algorithm makes a maximum of $O(klogn)$ mistakes.

Winnow algorithm:
1. Initialize $w_i = 1, \forall i \in A$.
2. If $\sum_i w_i x_i \geq n$, predict positive, else predict negative.
3. On a mistake, if the true label was positive, $w_i = 2w_i, \forall$ attributes i present in the example.
4. On a mistake, if the true label was negative, $w_i = \frac{w_i}{2}, \forall$ attributes i present in the example.
5. Keep repeating steps 2, 3, and 4 on all incoming examples on which we need to make predictions.

Claim 23.3.2 *Let P be the number of mistakes made on positive examples (the true label on the example is positive). $P \leq k \log_2 n$.*

Proof: Let us focus on any particular attribute j in the true expert. Whenever we make a mistake on an example containing the attribute j, we double the weight of that attribute. So we make atmost $log_2 n$ mistakes on positive examples containg attribute j before w_j becomes more than n, and from then on we do not make any mistakes on positive examples containg attribute j. Since we have atmost k such attributes in the true expert, we make atmost $klog_2 n$ mistakes before we stop making mistakes on positive examples. ■

Claim 23.3.3 *Let N be the number of mistakes made on negative examples (the true label on the example is negative). $N \leq 2P + 2$.*

Proof: Whenever we make a mistake on a positive example (the sum of weights on attributes in that example $< n$), the increase in sum of weights is atmost n. Likewise when we make a mistake on

a negative example (the sum of weights on attributes in that example $\geq n$), the reduction in sum of weights $\geq \frac{n}{2}$. The initial sum of weights of attributes is n. After P mistakes on positive examples, and N mistakes on negative examples, the sum of weights of all attributes $\leq n + nP - \frac{n}{2}N$. The total weight never falls below 0. So, $n + nP - \frac{n}{2}N \geq 0$. So, $N \leq 2P + 2$. ■

Theorem 23.3.4 *The bound on total number of mistakes is $O(k \log n)$ using the above two claims.*

23.4 r out of k Concept Class

Here the true expert again has k attributes, and if any r of them are present in the example, then the example is true, else false. In the next class, we will see how the Winnow algorithm (using a multiplicative factor of $(1+\epsilon)$ instead of 2) can be used to get a mistake bound of $O(krlogn)$ (using $\epsilon = \frac{1}{2r}$).