Scribe: Pratyush Tomar (2002439) Lecturer: Naveen Garg

Few lectures back we covered a simple algorithm for learning a monotone formula for which the number of mistakes to arrive at consistency model was n, where n is the number of different attributes. But if n is large and most of the attributes are irrelevant then this algorithm will be inefficient. If our target is an OR of r relevant attributes where r is lot smaller than n, then we can use Winnow algorithm which give us a bound of $r \log n$ mistakes. As the name suggest this algorithm is used to winnow away relevant from irrelevant attributes. Thus in this lecture we will look at Winnow algorithm, calculate mistake bound and then winnow for learning a majority-vote (k of r) function.

1 Winnow Algorithm

In this section we will describe the Winnow algorithm. There is a weight defined for each of the variable.

The Winnow Algorithm :

1. Initialize the weights $w_1, w_2, \ldots, w_n$ of the variables to 1.
2. Given an example $x = \{x_1, x_2, \ldots, x_n\}$, output + if

$$\sum_{i=1}^{n} w_i x_i \geq n$$

   else output $-$.
3. On a mistake
   (a) If the example was positive and algorithm predicts negative, then for each $x_i = 1$, double the value of w_i.
   (b) If the example was negative and algorithm predicts positive, then for each $x_i = 1$, halve the value of w_i.
4. Goto step 2.

2 Mistake bound of the Winnow Algorithm

In this section we will calculate the total number of mistakes by the algorithm.

Theorem 2.1 *The Winnow Algorithm makes at most $2 + 3r(\lceil \lg n \rceil)$ mistakes when the target concept is a disjunction of r variables.*

Proof: We will first bound the number of mistakes of 1st kind that is mistakes on the positive examples. Any mistake on a positive example must double at least one of the relevant weights in the target function for which $x_i = 1$, and a mistake on negative example will not change any of these weights, by definition of disjunction. Each of these weights can be doubled at most $\lceil \lg n \rceil$ times since only weights that are less than n can be doubled. Therefore, Winnow algorithm makes at most $r(\lceil \lg n \rceil)$ mistakes on positive examples.

Now we will bound the number of mistakes of 2nd kind that is mistakes on the negative examples. Initially the total weight summed over all variables is n. Suppose that a mistake occurs.

Each mistake on positive examples increase the total weight by at most n. In this case before doubling of weights we must had $\sum_{i=1}^{n} w_i x_i \rangle\ n$ and since only weights corresponding to $x_i = 1$ are doubled, the increase is less than n. While each mistake on negative example decrease the total weight by at least $\frac{n}{2}$. Because before halving we must had $\sum_{i=1}^{n} w_i x_i \geq n$. Since all the weights for which $x_i = 1$ are halved the decrease is atleast $\frac{n}{2}$. Also the total weight will never drop below zero. So we can get the following equation :

$$n + n(\text{number of mistakes of 1st kind}) - \tfrac{n}{2}(\text{number of mistakes of 2nd kind}) \geq 0$$

Solving this equation for mistakes of 2nd kind we get :

$$\text{number of mistakes of 2nd kind} \leq 2 + 2r(\lceil \lg n \rceil)$$

Hence, adding mistakes of both kind gives the theorem.

□

3 Winnow for Majority vote function

In this section we will see Winnow Algorithm for learning majority vote(k of r) function, where n is the number of attributes, r is relevant attributes and if any k out of these r are present then example is positive. Let us take an example of email messages and we need to classify SPAM or NOT SPAM. Thus if k keywords are present then we classify email as SPAM. To solve this problem we need to modify Winnow algorithm. We will multiply w_i by $(1+\epsilon)$ for mistake on positives and divide by $(1+\epsilon)$ for mistake on negatives.

Let, M_+ : number of mistakes on positive example(weight increases).
M_- : number of mistakes on negative example(weight decreases).

Now consider multiplying by $(1+\epsilon)$ as putting a chip on the weight, while dividing by $(1+\epsilon)$ as removing the chip.

So N = total number of chips on relevant variables. Now we know that the maximum value of the weight of relevant variable can be n, so number of chips is at most $log_{1+\epsilon}n$ on a relevant variable. Thus we can have following relationship :

$$\text{N} \leq r log_{1+\epsilon} n$$

Now for M_+ : N increases by k as we need to add atleast k chips for each positive example since there must have been k attributes since it was labeled positive.
M_- : N decreases by at most $k-1$ as for each negative example we remove at most $k-1$ chips since there must have been at most $k-1$ attributes since the example was labeled negative.

Thus we can get the 1st inequality :

$$kM_+ - (k-1)M_- \leq N \leq r log_{1+\epsilon} n$$

Let W = total weight of variables.

For M_+ : W increases by $\leq \epsilon n$.
M_- : W decreases by atleast $\frac{\epsilon n}{1+\epsilon}$. Also the total weight never goes below zero, thus we can get the 2nd inequality :

$$n + M_+ \epsilon n - \frac{M_- \epsilon n}{1+\epsilon} \geq W \geq 0$$

Thus solving above two inequalities for M_+ and M_- we get :

$$M_+,\, M_- = O(rk \log n) \text{ for } \epsilon = \frac{1}{2k}.$$

But if we don't know k then we can guess and double r but then cost is $O(r^2 * \log n)$.