Computational Learning Theory	**Spring semester, 1997**
Lecture 5: March 25, 1997	
Lecturer: Yishay Mansour	*Scribe: Ofir Friedman*

5.1 Experts and Boosting

Motivation

We have several experts for a certain task. We don't have any prior information on the reliability of the different experts. We want to use the experts gusses to predict, and hope to converge to the performance of the best expert. (Of course we cannot expect to perform better than the best expert, when in the general case.)

The online model

In each step we have full information about the past, and no information about the future. (In each step we know exactly wheather an expert was correct in his guess, or was wrong.)

The model includes:

1. N different strategies (experts).
2. In step t, the learner A, chooses a distribution P^t on the N strategies. ($\sum_{i=1}^{N} P^t(i) = 1$)
3. After each step, the learner receive the loss of each expert (denote by the vector l^t).
4. The loss of A at time t is $\sum_{i=1}^{N} P^t(i) l_i^t$.

Without the loss of generality we can assume that $l_i^t \in [0,1]$. Our goal is to build an algorithm A that minimize the loss of the learner. Of course the loss of A at least the loss of the best expert. The loss of the i-th expert is given by $L_i^t := \sum_{j=1}^{t} l_i^j$.

Examples

Guessing letters:
We have k letters and k experts, where expert i guess the i-th letter all the time. The loss is 1 if our letter matches the next letter, and 0 if it does not. It is easy to see, that in the worse case we will loose n in n successive tries. However, the best expert is loses at most $\frac{n}{k}$.

player 1 \ player2	I	II	III
I	0.5	1	0
II	0	0.5	1
III	1	0	0.5

Figure 5.1: 2 players game

Game theory:
Two players play a game. player 1 chooses a row, player 2 chooses a column. Each of the players have 3 different strategies (Experts). The loss of player 1, is as shown on the table above.
Given that player 2 chooses his strategies in random (nature), no matter what will be the strategy of player 1, he will loose $\frac{n}{2}$ in n games.
If, on the other hand, player 2 has one fixed strategy, then player 1 must choose his best corresponding strategy.

Best Expert Algorithm

In step t, we choose the guess of the expert that has the minimal L_i^t.

Theorem 5.1 *Let L_A^T be the loss of A in time T. Then:*

$$L_A^T \leq N(\min_i L_i^T + 1)$$

where N is the number of experts.

Proof: Let l_i^t be the number of times that A chose expert i and lost, until the time t. clearly, $\forall i, l_i^t \leq L_i^t$.
Also, it is easy to see, that:

$$\forall t, j \ 0 \leq l_j^t - \min_i(L_i^t) \leq 1$$

Since otherwise, in the previous step we chose expert j, when we shouldnt have. Now:

$$L_A^T = \sum_{j=1}^{N} l_j^T \leq \sum_{j=1}^{N} \min_i(L_i^T) + 1 \leq N(\min_i(L_i^T) + 1)$$

■

¿From the guessing letters example, one can see that this bound is tight.

5.1.1 Hedge(β)

This is a much better algorithm. In each step t, the algorithm maintains a positive weight vector w^t for the experts.
Now, we define the probability vector $P^t = \frac{w^t}{\sum_{i=1}^{N} w_i^t}$.
We initialize for every ı, $w_i^1 = \frac{1}{N}$. After we receive l^t, we updates the weights as follows:

$$w_i^{t+1} = w_i^t U_\beta(l_i^t)$$

Where $U_\beta : [0,1] \to [0,1]$, such that for any $r, \beta \in [0,1]$, we have

$$\beta^r \leq U_\beta(r) \leq 1 - (1-\beta)r.$$

Let us try and motivate the update rule. Assume that $U_\beta(r) = \beta^r$, then:

$$w_i^t = w_i^{t-1}\beta^{l_i} = \ldots = w_i^1\beta^{L_i^{t-1}}.$$

Note, that when an expert is wrong, his weight is going down, and he will have a smaller effect on the following guesses. If we have one (or several) good expert, his weight in the final guess will dominate, and the performance of the algorithm will be close to that best expert. Unlike the previous algorithm, the predictions will not be dominated by a single expert at each step but by the set of almost bests experts.

Analysis of Hedge(β)

Lemma 5.2 *for every series of loses,* $l^1 \ldots l^T$,

$$ln(\sum_{i=1}^{N} w_i^{T+1}) \leq -(1-\beta)L_{Hedge(\beta)}$$

Proof: Choose a certain time t, then we have,

$$\sum_{i=1}^{N} w_i^{t+1} = \sum_{i=1}^{N} w_i^t U_\beta(l_i^t) \leq \sum w_i^t(1-(1-\beta)l_i^t) = (\sum_{j=1}^{N} w_j^t)(1-(1-\beta)P^t l^t)$$

(For the last equality we use the identity $w_i^t = P_i^t(\sum w_j^t)$).

We will take the log of the expression, and using the inequality $ln(1-x) \leq -x$, we have:

$$ln(\sum w_i^{t+1}) \leq ln(\sum w_j^t) + ln(1-(1-\beta)P^t l^t) \leq ln(\sum w_i^t) - (1-\beta)P^t l^t$$

Meaning that the growth of the log of the weight in each step, is bounded by $-(1-\beta)P^t l^t$. Therefore,

$$ln(\sum w_i^{t+1}) \leq ln(\sum w_i^1) - (1-\beta)\sum P^t l^t = -(1-\beta)L_{Hedge(\beta)}$$

■

¿From the Lemma, we get :

$$L_{Hedge(\beta)} \leq \frac{-ln(\sum_{i=1}^{N} w_i^{t+1})}{1-\beta}$$

Since the weights are positive for any i:

$$\sum w_i^{T+1} \geq w_j^{T+1} \geq w_j^1 \beta^{L_j} = \frac{1}{N}\beta^{L_j}$$

Therefore:

$$\forall i \quad L_{Hedge} \leq \frac{-ln(\frac{1}{N}\beta^{L_i})}{1-\beta} = \frac{lnN + L_i ln\frac{1}{\beta}}{1-\beta}$$

If we set $\beta = \frac{1}{2}$, we get the bound of $2lnN + (ln2)min_i(L_i)$, which is much better that what we had (A fairly weak dependency on th number of experts). We want to choose β to get an even better bound.

Lemma 5.3 *For $0 \leq L \leq L'$ and $0 \leq R \leq R'$ we sould choose $\beta = g(\frac{L'}{R'})$ when $g(z) = \frac{1}{1+\sqrt{\frac{2}{z}}}$ and then*

$$\frac{-Lln\beta + R}{1-\beta} \leq L + \sqrt{2L'R'} + R$$

Using the above lemma, we can choose $L' = maximumlose$ and $R' = lnN$ and get that for $\beta = g(\frac{L'}{lnN}$ we have that

$$L^A \leq min(L_i) + \sqrt{2L'lnN} + lnN$$

which means that there is only additive.

5.2 Boosting

In the PAC model that we define last time, there is a distribution D on X. The random examples $< x, c(x) >$ are drawn such that x is distributed according to D and $c \in C$ is the target function.

The goal is to find a hypothesis $h \in H$ such that $error(h, c) \leq \epsilon$, with probability $1 - \delta$. This is a *strong learning model*, since ϵ can be as small as we want.

We define a *weak learning model* by replacing the accuracy requirement by :

$$error(h, c) \leq \frac{1}{2} - \gamma$$

.

Intuitively, it is easier to find hypothesis that is correct only 51 percent of the time, rather than a hypothesis that is correct 99 percent of the time. We show, that if a concept class has a weak learning algorithm, then there is a PAC learning algorithm for the class.
We will change the definition : $error(h, c) = E_{X\ D}[|h(x) - c(x)|]$, when $c(x) \in \{0,1\}$ and $h(x) \in [0, 1]$, meaning that the hypothesis is probabilistic, insead of deterministic.

First, let us define our prediction hypothesis. For any set of h function from H, we have a hypothesis which is their majority, i.e. $MAJ(h_1 \ldots h_T)$, for $h_i \in H$. Our PAC learning algorithm will work as follows.
We will take sample of m points, find a hypothesis $h^A = MAJ(h_1 \ldots h_T)$, which is consistant with the sample. If we can gurantee that,

$$size(h_A) = Tsize(h_i) << m,$$

then, by *Occam Razor Theorem*, we can PAC learn the class.

We will use the weak hypothesis as follows. We will take the sample of m points with a uniform distribution and look for a weak hypothesis from H. If we find the weak hypothesis has error less than $\frac{1}{m}$, we are done, since it has to be consistent. Else, we change the distribution on the m points, and get another weak hypothesis h_i. At the end we return the majority of all the weak hypothesys h_i.

First, we will assume that we know a value γ, such that the errors $\epsilon_1 \ldots \epsilon_T$ are bounded by $\frac{1}{2} - \gamma$, where $\epsilon_i = error(h_i, c)$. The "experts" will be the points. The loss of the expert i in step t will be $l_i^t = 1 - |h_t(x_i) - c(x_i)|$ (Note that the loss is 1 if h_t is correct on x_i and 0 if it is incorrect).
We fix $\beta = 1 - \gamma$ and our final hypothesis $h^A = MAJ(h_1 \ldots h_T)$.
We need to show how the error of h^A drops when T increases.

Define, $S = \{i|h^A(x_i) \neq c(x_i)\}$, i.e. the set of points on which our final hypothesis errors. For $i \in S$ let

$$\frac{L_i}{T} = \frac{1}{T}\sum_{t=1}^{T} l_i^t = 1 - \frac{1}{T}\sum_{t=1}^{T} |c(x_i) - h_t(x_i)| = 1 - |c(x_i) - \frac{1}{T}\sum_{t=1}^{T} h_t(x_i)| \leq \frac{1}{2},$$

¿From what we saw before on $Hedge(\beta)$,

$$T(\frac{1}{2} + \gamma) \leq \sum P^t l^t \leq \frac{-ln(\sum_{i \in S} D(i)) + (\gamma + \gamma^2)\frac{T}{2}}{\gamma}.$$

Since $-ln\beta = -ln(1-\gamma) \leq \gamma + \gamma^2$

The probability of error is $\epsilon = \sum_{i \in S} D(x_i)$ and thus:

$$\epsilon \leq e^{\frac{-T\gamma^2}{2}}$$

We need that $\epsilon \leq \frac{1}{m}$ and get: $T \geq \frac{2ln(m)}{\gamma^2}$

We achived our goal. We showed that if a class has a weak learning algorithm, it has a strong learning algorithm. From a practical point of view it is very unforetunate that we need to know γ in advance, to run the algorithm (we need it in order to define β).

5.3 The AdaBoost

The advantage of this algorithm is that we don't need to know γ in advance, and that we don't need a uniform bound on all the weak learners. In addition, we will be able to show better error bounds. Ada Boost in each iteration, we use a different β_t depend on ϵ_t, which is the error of the last weak hypothesis.

Initialization : $w_i^1 = D(x_i)$
The algorithm AdaBoost (in the t step) does:

1. $P^t = \frac{w^t}{\sum w_i^t}$

2. Call the *weak learner*, with P_t and get hypothesis h_t.

3. Compute the error of h_t on the sample : $\epsilon_t = \sum_{i=1}^{m} P^t(i)|h_t(x_i) - c(x_i)|$

4. Fix $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

5. Let $w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-c(x_i)|}$

The output will be $h^A(x) = 1 \iff \sum_{t=1}^{T}(log\frac{1}{\beta_t})h_t(x_i) \geq \frac{1}{2}\sum log(\frac{1}{\beta_t})$

Theorem 5.4 *Suppose that the weak learner geneated hyphotesis with errors* $\epsilon_1 \dots \epsilon_T$ *then the error* ϵ *of* h^A *is bounded by:*

$$\epsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}$$

Proof: Denote $l_i^t = 1 - |h_t(x_i) - c(x_i)|$. Since l_i^t is 0 when there is an error and 1 otherwise, it follows that, $1 - \epsilon_t = P^t l^t$.
When we analized $Hedge$, we showed that

$$\sum_{i=1}^{m} w_i^{t+1} \leq (\sum w_i^t)(1 - (1-\beta_t)l^t P^t)$$

and thus

$$\sum_{i=1}^{m} w_i^T \leq \prod_{t=1}^{T}(1 - (1-\beta_t)(1-\epsilon_t)),$$

The hypothesis h_A will make an error, only if

$$\prod_{t=1}^{T} \beta_t^{-|h_t(x_i)-c(x_i)|} \geq (\prod_{t=1}^{T} \beta_t)^{-\frac{1}{2}}$$

(because $c(x_i) \in \{0,1\}$). The final weight of the point i, is :

$$w_i^{T+1} = D(x_i) \prod_{t=1}^{T} \beta_t^{1-|h_t(x_i)-c(x_i)|} \geq (\prod_{t=1}^{T} \beta_t)^{\frac{1}{2}} D(x_i)$$

When we sum over all points we have:

$$\sum_{i=1}^{m} w_i^{T+1} \geq \sum_{i \in S} D(x_i)(\prod_{t=1}^{T} \beta_t)^{\frac{1}{2}} = \epsilon \sqrt{\prod_{t=1}^{T} \beta_t},$$

We already bounded the sum of the weigths, and thus:

$$\epsilon \leq \prod_{t=1}^{T} \frac{1-(1-\epsilon_t)(1-\beta_t)}{\sqrt{\beta}_t}$$

Since every element in the product is positive, we can minimize each element, by choosing $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$, and get:

$$1-(1-\epsilon_t)(1-\beta_t) = 2\epsilon_t$$

This implies that our error bound is:

$$\epsilon \leq \prod 2\sqrt{\epsilon_t(1-\epsilon_t)}.$$

■

Assuming that for all t's we have $\epsilon_t = \frac{1}{2} - \gamma$, we get the bound:

$$\epsilon \leq (1-4\gamma^2)^{\frac{T}{2}} \leq e^{-2\gamma^2 T}$$

Note however, that AdaBoost takes into account the accuracy of different hypothesis, and therefore in practise it might set much better bounds.