**CS-621 Theory Gems** September 19, 2012

# Lecture 1

*Lecturer: Aleksander Mądry* *Scribes: Alexandre Duc and Alina Dudeanu*

## 1 How to Get Rich (If You Have Good Advice)

The main topic of this lecture is the learning-from-expert-advice framework. Our goal here is to be able to predict, as accurately as possible, a sequence of events in a situation when our only information about the future is coming from recommendations of a set of "experts". The key feature (and difficulty) of this scenario is that most – if not all – of these experts (and thus their recommendations) might be unreliable or even adversarial. Therefore, even if one of them is consistently providing fairly accurate predictions, we need to be able to efficiently extract this advice from all the misleading recommendations that we are receiving.

To cope with this task, we will develop a family of algorithms based on so-called *multiplicative weights update method*. We will show that these algorithms allow us to achieve close-to-optimal *regret minimization* bounds. That is, provided the sequence of predictions is sufficiently long, our prediction performance will be close to the performance of the expert that was best in the hindsight.

Finally, we will see that this algorithmic approach also applies to a certain natural generalization of the above model.

As we will see later in the course, the ideas we discuss here have wide-spread applications throughout Computer Science – most notably, in Machine Learning, Optimization, and Game Theory.

## 2 The Stock Market Model

We will model our framework as a very simplified version of stock market prediction. In this version, we view the stock market as a single evolving index $X$ – one can think of it as, e.g., the value of the Dow Jones. This index starts at some value, say $X(0) = 0$ and each day (round) $t$ it can either go or down up by one (i.e., either $X(t+1) = X(t) - 1$ or $X(t+1) = X(t) + 1$). Our task is to predict at the beginning of each day which one of these two possibilities will occur. We are interested in minimizing our total number of prediction mistakes over a fixed (but very large) number $T$ of days this games repeats.

It is not hard to see that if we are not equipped with any knowledge on the behavior of $X$, we cannot hope for any gain over random guessing (that, in expectation, would make us mis-predict every other round). In fact, in the (not that unrealistic) case when we do not have access to true randomness and the market is completely adversarial to us, one might end up mis-predicting in every round.

So, to make this model interesting, we introduce a set of $n$ "experts" (or sources of information) such that at the beginning of each round $t$, each one of them provides us with his/her "prediction" of the behavior of $X$ in that round. (For instance, one can think of these experts as financial advisors from CNN, Wall Street Journal, etc.)

Unfortunately, in general, we have no guarantee that these predictions are correct. That is, the experts might be actually feeding us random (or even adversarial) recommendations. So, our goal is to develop an algorithm that can utilize this information to make fairly good prediction, as long as, at least one (initially unknown to us) expert is consistently providing good advice.

## 3 Deterministic Algorithms

We begin with investigation of deterministic solutions for this problem.

## 3.1 The Halving Algorithm

We start with a simpler scenario in which we know that at least one of the experts provides a perfect advice (i.e., his/her advice is always correct).

Probably the most natural first approach would be to just always go with the majority vote of the experts. Unfortunately, one can easily see that when all but one experts are always providing incorrect predictions this approach fails miserably.

Interestingly, if we just fix the above idea by discarding the prediction of all the experts that already were wrong in the past, the performance of the resulting algorithm improves dramatically. More precisely, we consider the following *Halving Algorithm*:

1. At the beginning, each expert $i$ is in the pool of "trustworthy" experts denoted by $S$
2. In each round $t$:
   (a) We follow the majority vote of all the experts in $S$ (breaking ties arbitrarily)
   (b) After seeing $X(t)$, we remove from $S$ all the experts that made an incorrect prediction in this round

By our assumption that a perfect expert exists, we know that the set $S$ is never empty. Furthermore, at the end it contains all the experts that were always correct. The performance of this algorithm is given by the following lemma.

**Lemma 1** *If there is a perfect expert, Halving Algorithm will do at most $m \leq \log_2(n)$ mistakes.*

**Proof** Every time the Halving Algorithm makes a mistake, we remove at least half of the experts from $S$. Since at the beginning $S$ contains $n$ experts and it can never be empty, the bound of $\log_2(n)$ follows. ■

Note that somewhat interestingly, the number of mistakes the halving algorithm does is independent of the length of the sequence $T$, it depends only on the number of experts. Also, one can see that the bound of $\log_2 n$ is essentially tight for deterministic algorithms.

## 3.2 The Weighted Majority Algorithm

Clearly, the assumption that there always exists a perfect expert is pretty strong and rather unrealistic. After all, in the real world, nobody is perfect. Let us consider then what happens when even the best-in-the-hindsight expert makes $m^*$ prediction mistakes (and we do not know the value of $m^*$).

A tempting approach here would be to simply adopt the Halving Algorithm. Namely, we could consider the following *Iterated Halving Algorithm*:

1. At the beginning, each expert $i$ is in the pool $S$.
2. In each round $t$:
   (a) We follow the majority vote of all the experts in $S$ (breaking ties arbitrarily)
   (b) After seeing $X(t)$, we remove from $S$ all the experts that made an incorrect prediction in this round
   (c) If the pool $S$ becomes empty, we reset it by putting all the experts back into it

**Lemma 2** *The number of mistakes $m$ made by the Iterated Halving Algorithms is at most $(m^*+1)(1+\log_2 n)-1$.*

**Proof** Note that whenever the set $S$ is reset, it must be the case that each expert made at least one mistake since the last reset. Therefore, there can be at most $m^*$ resets overall.

Now, using the reasoning from the proof of Lemma 1, we can see that in each period before a reset it will take at most $1 + \log_2 n$ of mistakes of the algorithm till the set $S$ is empty (and thus the reset is triggered). On the other hand, after the last reset, there is at most $\log_2 n$ mistakes total (as otherwise, the set $S$ would become again empty). So, the total number of mistakes of the algorithm can be bounded by:

$$m^*(1 + \log_2 n) + \log_2 n = (m^* + 1)(1 + \log_2 n) - 1,$$

as desired. ■

Although the bound obtained above is proportional only to the number $m^*$ of mistakes of the best expert (and not to $T$), as well as, depends only logarithmically on the number of experts, there is still much room for improvement.

For instance, one shortcoming of the above algorithm is that each reset is completely discarding the knowledge we accumulated till that moment. In particular, we do not differentiate there between an expert that did only a few mistakes up until now and a one that was constantly giving incorrect predictions. Also, given that now even the best expert can make a mistake from time to time, having in each phase a sharp disqualification due to one mistakes might be suboptimal.

These considerations lead to an idea in which instead of keeping a partition into "trustworthy" and (temporarily) "not trustworthy" experts, we will assign a weight to each expert that expresses our confidence in his prediction based on his/her performance so far. Whenever we have to make a prediction, we just go with the *weighted* majority vote (i.e., a one that is proportional to weights) and then penalize the experts that were wrong by appropriately decreasing their weights.

This idea gives rise to the following *Weighted Majority Algorithm*

1. At the beginning, each expert $i$ is assigned a weight $w_i = 1$.
2. In each round $t$:
   (a) We follow the weighted majority vote of the experts
   (b) After seeing $X(t)$, we decrease by $\frac{1}{2}$ the weight $w_i$ of each expert $i$ that made a mistake this round

**Lemma 3** *The number of mistakes $m$ of the Weighted Majority algorithm is at most $2.4(m^* + \log_2 n)$*

**Proof** Let $w_i^t$ be the weight of experts $i$ after round $t$. We look at the potential function $W^t = \sum_i w_i^t$.

Clearly, we have that $W^0 = n$ and $W^t$ can only decrease with time.

We want to study how much the potential function decreases whenever the algorithm makes a mistake. By definition of the algorithm, if we make a mistake in round $t$ then the total weight of the experts that were incorrect is at least $W^{t-1}/2$. As their weight is cut in half, we must have

$$W^t \leq \frac{3}{4} W^{t-1}.$$

Thus we can conclude that after the last round $T$

$$W^T \leq \left(\frac{3}{4}\right)^m n.$$

On the other hand, if $i^*$ is the best expert then at the end we will have that

$$w_{i^*}^T = \left(\frac{1}{2}\right)^{m^*}.$$

As obviously $W^T \geq w_{i^*}^T$, we can conclude that

$$\left(\frac{1}{2}\right)^{m^*} \leq \left(\frac{3}{4}\right)^{m} n.$$

Taking logarithms of both side, we get

$$m^* \log_2 \frac{1}{2} \leq m \log_2 \frac{3}{4} + \log_2 n$$

and, since $-1/\log_2 \frac{3}{4} \approx 2.4$, we conclude that

$$m \leq 2.4(m^* + \log_2 n),$$

as desired. ■

Note that the obtained performance surpasses the one of the Iterated Halving Algorithm already when $m^* \geq 3$ (which is a very typical case). Most importantly, we are only a constant factor away from $m^*$ – that is, the $\log_2 n$ term appears only additively (instead of multiplicatively). A natural question at this point is whether the constant 2.4 in front of $m^*$ can be improved. As it turns out that it can be improved a bit more, but, as long as, our algorithms are deterministic, the constant in front of $m^*$ can never be smaller than 2. Fortunately, as we will see shortly, once we employ randomness, further improvement is possible.

# 4 Randomized Algorithms

Now that we considered deterministic algorithms, let us have a look at randomized one to see if they can perform better.

## 4.1 Randomized Halving Algorithm

Let us first come back to the case where the best expert does not do any mistake, i.e., $m^* = 0$. We will now try to apply randomization to improve the guarantee given by the Halving Algorithm. The new algorithm that we employ is called the *Randomized Halving Algorithm* since it is a variation of the Halving Algorithm in which majority selection is replaced by a random choice among the experts that are still in the pool.

More precisely, we consider the following algorithm:

1. At the beginning, each expert $i$ is in the pool of "trustworthy" experts denoted by $S$
2. In each round $t$:
   (a) Choose a random expert from $S$ and follow his/her prediction
   (b) After seeing $X(t)$, we remove from $S$ all the experts that made an incorrect prediction in this round

**Lemma 4** *The expected number of mistakes $E[m]$ of the Randomized Halving Algorithm is at most $\ln(n)$.*

**Proof** Let $F_t$ be the fraction of *alive* experts (i.e., the ones still in the pool $S$) that are wrong in round $t$. This means that at round $t$, we will make a mistake with probability $F_t$. Hence, summing over all the rounds, we get that the expected number of mistakes we will do is

$$E[m] = \sum_t F_t \ . \tag{1}$$

Now, it is not hard to see that the fraction of the *initial* experts that are still alive after $t$ steps is

$$P_t \coloneqq \prod_t (1 - F_t) \geq \frac{1}{n} ,$$

where the last inequality holds as the best (perfect) expert will never be removed from the pool.

Taking a natural logarithm of this equation, we get

$$\sum_t \ln(1 - F_t) \geq -\ln(n) . \tag{2}$$

Recall that the Taylor expansion of $\ln(1 - x)$ is $-\sum_{n=1}^{\infty} \frac{x^n}{n}$ for $0 \leq x < 1$. Hence, $\ln(1 - x) < -x$ for $0 \leq x < 1$. Using this inequality and Equation (2), we get

$$\sum_t F_t \leq \ln(n) .$$

Combining with Equation (1) gives us the required result, i.e.,

$$E[m] \leq \ln(n) .$$

■

Note that $\ln n \approx 0.7 \log_2 n$. So, the performance of this algorithm is strictly better than the best possible performance of a deterministic solution. Of course, the price of that is that now the mistake guarantee holds only in expectation.

## 4.2 Randomized Weighted Majority Algorithm

We proceed now to developing a randomized algorithm for the general case, i.e., a one in which the best expert is making $m^*$ mistakes. Once again, our algorithm will be a simple modification of its deterministic equivalent. We update the Weighted Majority Algorithm by making it take randomized weighted majority vote (instead of a deterministic one). Furthermore, instead of decreasing the weights of incorrect experts by a factor $1/2$ we consider a more general update by a factor of $(1 - \varepsilon)$, where $\varepsilon$ is a tuning parameter of the algorithm. (Clearly, taking $\varepsilon = 1/2$ recovers the original update.)

Formally, the *Randomized Weighted Majority Algorithm* works as follows

1. At the beginning, all experts have weight 1.
2. In each round $t$:
   (a) We choose one of the experts at random (with probability proportional to his/her weight) and follow his/her advice
   (b) After seeing $X(t)$, we decrease by $(1 - \varepsilon)$ the weight $w_i$ of each expert $i$ that made a mistake this round

The performance of this algorithm is given by the following lemma.

**Lemma 5** *If $0 < \varepsilon \leq \frac{1}{2}$ and $E[m]$ is the expected number of mistakes of the Randomized Weighted Majority Algorithm then*

$$E[m] \leq (1 + \varepsilon)\, m^* + \frac{\ln n}{\varepsilon} . \tag{3}$$

**Proof** The proof is very similar to the one of Lemma 4. Let $F_t$ be the *weighted* fraction of experts that are wrong in round $t$. Let $w_i^t$ be the weight of expert $i$ at the end of round $t$ and let $W^t \coloneqq \sum_i w_i^t$.

Obviously, $W^0 = n$. Also, as in the previous proof, one has that $E[m] = \sum_t F_t$. Now, note that

$$W^t \leq n(1-\varepsilon F_1)(1-\varepsilon F_2)\dots(1-\varepsilon F_t) = n\prod_{t\leq T}(1-\varepsilon F_t)\ .$$

Let $i^*$ be the index of the best expert. Since he cannot make more than $m^*$ mistakes, we have

$$w_{i^*}^T \geq (1-\varepsilon)^{m^*}\ .$$

We get

$$(1-\varepsilon)^{m^*} \leq w_{i^*}^T \leq W^T \leq n\prod_{t\leq T}(1-\varepsilon F_t)$$

Taking the natural logarithm of both sides gives us

$$m^*\ln(1-\varepsilon) \leq \ln n + \sum_{t\leq T}\ln(1-\varepsilon F_t)$$

which can be rewritten as (using the same Taylor expansion approximation as before)

$$m^*\ln(1-\varepsilon) \leq \ln n - \sum_{t\leq T}(\varepsilon F_t) = \ln n - \varepsilon E[m].$$

This, in turn, gives us that

$$E[m] = \sum_{t\leq T} F_t \leq -\frac{\ln(1-\varepsilon)}{\varepsilon}m^* + \frac{\ln n}{\varepsilon} \leq (1+\varepsilon)\,m^* + \frac{\ln n}{\varepsilon},$$

since, again, due to Taylor expansion $\ln(1-\varepsilon) \geq -\varepsilon - \varepsilon^2$ for $0 \leq \varepsilon \leq \frac{1}{2}$, we have $-\ln(1-\varepsilon)/\varepsilon \leq 1+\varepsilon$. ■

Note that, as $\varepsilon$ becomes smaller, our multiplicative overhead over $m^*$ is approaching 1, but the additive term is growing. So, we usually want to choose $\varepsilon$ in a way that balances out these two effects. Also, as we typically expect that $m^*$ grows as the total number of turns $T$ becomes larger, one can see that for sufficiently large $T$ the "average" number of mistakes of this algorithm is close to optimal. Finally, we point out that if one changes the weighted update factor in (deterministic) Weighted Majority Algorithm from $1/2$ to $(1-\varepsilon)$ in the same manner as it was done here, then one can get the multiplicative factor in the mistake bound of this algorithm (cf. Lemma 2) to be arbitrarily close to 2 (while having the additive term grow accordingly).

# 5 General Learning-From-Expert-Advice Framework

It turns out that the ideas we explored above extend in a simple way to a much more general (and very useful) variation of learning-from-expert-advice framework.

In this variation, we again have $n$ "experts" (although now, one might view them as choices/options) and we are playing $T$ rounds of the following game:

1. In each round $t$ we need to choose a *convex combination* $p_1^t, \dots, p_n^t$ of experts
2. Once we have made our choice, a "loss" $l_i^t \in [-\rho, \rho]$ is revealed for each expert $i$ – here, $\rho$ is a parameter of the whole sequence and is known to the player upfront

3. Our loss in round $t$ is $\sum_i p_i^t \; l_i^t$

Our goal is to devise a strategy for choosing the convex combination at each step, so that the total loss we incur is not much worse than the loss of the best-in-the-hindsight expert.

Note that in this framework we allow $l_i^t$ to be negative (i.e., they can correspond to gains). Also, one should note that the convex combinations can be directly interpreted as probabilities. So, by taking a random choice according to them in each round, one can get the same performance bound in expectation. (This is important in some applications where taking a convex combination of the available experts is not feasible.)

Now, it is not hard to see that this setting generalizes our stock market model. One just need to take $l_i^t$ to be 1 if the expert $i$ makes a wrong prediction in round $t$ and $l_i^t$ to be 0 otherwise. Clearly, our total (expected) loss will be equal to the expected number of mistakes we make. What is interesting, however, is that the algorithms for our restricted setting can be easily extended to this new framework.

More precisely, the following algorithm (called *Multiplicative Weights Update Algorithm*) is very useful here.

1. At the beginning, all experts have weight $w_i^0 = 1$.
2. In each round $t$:
   (a) We choose $p_i^t := \frac{w_i^{t-1}}{W^{t-1}}$ for each $i$, where $W^t := \sum_i w_i^t$ (i.e., we take each expert proportionally to his/her weight)
   (b) After seeing all $l_i^t$, we modify the weight of each expert $i$ to be $w_i^t = (1 - \frac{\varepsilon}{\rho} l_i^t) w_i^{t-1}$

Again, the algorithm is parametrized by a tuning parameter $\varepsilon$ and this time we want $\varepsilon \leq 1/2$. Also, one should note that by definition of $\rho$ we have that $(1 - \varepsilon) \leq (1 - \frac{\varepsilon}{\rho} l_i^t) \leq (1 + \varepsilon)$.

Observe that in the case when $l_i^t$ is either 0 or 1 and we make randomized choices of the experts (instead of taking convex combinations), this algorithm is exactly the Randomized Weighted Majority Algorithm. Also, the approach to analyzing this new algorithm (as well as, the flavor of its performance bound) is very similar.

**Theorem 6** *For any $0 < \varepsilon \leq 1/2$ and any expert $i$, the total loss $l_{MWU}$ of the Multiplicative Weights Update Algorithm is such that*

$$l_{MWU} \leq \sum_t l_i^t + \varepsilon \sum_t |l_i^t| + \frac{\rho \ln n}{\varepsilon}.$$

Note that as this theorem allows us to compare our performance with respect to an arbitrary expert then in particular it has to hold for the best one among them. Again, if we look at the case of all $l_i^t$ being either 0 or 1 (with randomized choices of the experts) we essentially recover the guarantees of Randomized Weighted Majority Algorithm (cf. Lemma 5).

**Proof** As already mentioned, the proof is just a simple generalization of the proof of Lemma 5.

Clearly, $W^0 = n$. Now, in each round $t$ we have

$$W^t = \sum_i w_i^t = \sum_i (1 - \frac{\varepsilon}{\rho} l_i^t) w_i^{t-1} = W^{t-1} - \frac{\varepsilon W^{t-1}}{\rho} \sum_i p_i^t l_i^t = (1 - \frac{\varepsilon}{\rho} l_{MWU}^t) W^{t-1},$$

where $l_{MWU}^t$ is the loss of the algorithm in round $t$. So, we have that

$$W^T = \prod_t (1 - \frac{\varepsilon}{\rho} l_{MWU}^t) n.$$

Now, we can always lowerbound $W^T$ by $w_i^T$ obtaining that

$$\prod_t (1 - \frac{\varepsilon}{\rho} l_i^t) = w_i^T \leq W^T \leq \prod_t (1 - \frac{\varepsilon}{\rho} l_{MWU}^t) n.$$

Taking logarithms of both sides and using the fact that for $x \in [-\frac{1}{2}, \frac{1}{2}]$,

$$-x - x^2 \leq \ln(1 - x) \leq -x,$$

we get that

$$-\frac{\varepsilon}{\rho} \sum_t (l_i^t + \frac{\varepsilon}{\rho} (l_i^t)^2) \leq -\frac{\varepsilon}{\rho} \sum_t l_{MWU}^t + \ln n = -\frac{\varepsilon}{\rho} l_{MWU} + \ln n.$$

Rearranging the terms, multiplying both sides by $\frac{\rho}{\varepsilon}$, and using the fact that $(l_i^t)^2 \leq \rho |l_i^t|$, we obtain

$$l_{MWU} \leq \sum_t l_i^t + \varepsilon \sum_t |l_i^t| + \frac{\rho \ln n}{\varepsilon},$$

as desired. ■