Lecture 12 (29.05.2013)

Scribe: Marcin Mucha
Lecturer: Marcin Mucha

Multiplicative Weights Update

1 Introduction

In this lecture we will focus on the learning-from-expert-advice framework and the Multiplicative Weights Update (MWU) algorithm. We start with a very simple setting where we are, essentially, guessing the outcomes of a sequence of coin-flips (think: stock market). One cannot hope to beat the market without any model of the market behaviour. Our knowledge will come from *experts*, each of which makes predictions himself. Of course, the experts could be wrong (and often are in case of real stock market predictions) and might even be intentionally misleading us (and often are...). Therefore we will not even attempt to optimize the absolute performance of our strategy, but instead try to perform well compared to the best expert in hindsight. In a sense, we are asking how many mistakes does it cost to learn which expert (or in general which combination of experts) is the best one.

The initial coin-flipping setting is very simple but, surprisingly, it captures the essence of the problem. We later generalize it to the standard learning-from-expert-advice framework, and most of the ideas developed for the simple setting apply there as well. Interestingly, the general framework has several powerful applications. The two we discuss here are Von Neumann's Min-Max Theorem and solving LPs approximately.

2 The Basic Model

In the basic model we are trying to guess a sequence of coin-flips $x_1, \ldots, x_T$. We make predictions one by one, and before each prediction we can consult a set of n experts (the set of experts is fixed throughout the whole run) and base our prediction on their choices. After we make our prediction, we learn the actual outcome and proceed to the next prediction. Our goal is minimize the number m of mistakes made by our algorithm as a function of the number m^* of mistakes of the best expert in hindsight.

3 Deterministic Algorithms

Let us first consider the case where there is a perfect expert, i.e. $m^* = 0$. In this case, the following algorithm does very well.

Algorithm 1: The Halving Algorithm

```
Put all experts in a set S of trustworthy experts;
foreach round t do
    Follow the majority of experts in S;
    Remove from S all experts wrong in round t;
```

Theorem 1. *If there is a perfect expert, the Halving Algorithm makes at most $m \leq \log_2 n$ mistakes.*

Proof. Each time the algorithm makes a mistake, the size of S is halved. Since there is a perfect expert, S remains non-empty, and so $m \leq \log_2 n$. □

If we allow $m^* > 0$, the above approach fails, because eventually S becomes empty. We can easily fix this as follows.

Algorithm 2: The Iterated Halving Algorithm

```
Put all experts in a set S of trustworthy experts;
foreach round t do
    Follow the majority of experts in S;
    Remove from S all experts wrong in round t;
    If S becomes empty, put all experts back in;
```

Theorem 2. *The Iterated Halving Algorithm makes at most $(m^* + 1)(1 + \log_2 n) - 1$ mistakes, where m^* is the number of mistakes made by the best (in hindsight) expert.*

Proof. Using the same reasoning as before, we can show that going from $S = \{1, \ldots, n\}$ to S empty, we make at most $1 + \log_2 n$ mistakes (the extra mistake is there to account for going from $|S| = 1$ to $|S| = 0$). This can happen at most m^* times. After the best expert makes his m^* mistakes, we can make one last run, but this time only down to $|S| = 1$. Overall we get

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

□

This is not a bad result, as we clearly need to make at least $\max(m^*, \log_2 n)$ (prove it!). But it would be even better if we could get $m = O(m^* + \log_2 n)$. Clearly, there is place for improvement in the above algorithm, as it forgets everything after each reset, which seems suboptimal. The following approach attempts to accumulate the knowledge throughout the whole run using a softer distinction between trustworthy and non-trustworthy experts. Instead of dividing the experts into two groups, we instead use weights.

Algorithm 3: The Weighted Majority Algorithm

```
Set w_i := 1 for all experts i;
foreach round t do
    Follow the weighted majority vote of all experts;
    foreach expert i that turned out wrong do
        w_i := w_i(1 − ε);
```

Theorem 3. *The Weighted Majority Algorithm makes at most $2(1 + \varepsilon)m^* + \frac{2 \ln n}{\varepsilon}$ mistakes.*

Proof. Let $W^0 = n$ be the initial total weight, and W^t be the total weight after t rounds. If we make a mistake in round t, we have

$$W^t \leq \left(1 - \frac{\varepsilon}{2}\right) W^{t-1}.$$

Therefore, at the end we have

$$W^T \leq \left(1 - \frac{\varepsilon}{2}\right)^m n.$$

On the other hand

$$W^T \geq (1-\varepsilon)^{m^*},$$

since this is the weight of the best expert. Therefore

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

Taking logarithms of both sides we get

$$m^* \ln(1-\varepsilon) \leq m \ln\left(1 - \frac{\varepsilon}{2}\right) + \ln n,$$

and so

$$m \leq \frac{\ln(1-\varepsilon)}{\ln\left(1-\frac{\varepsilon}{2}\right)} m^* - \frac{\ln n}{\ln\left(1-\frac{\varepsilon}{2}\right)}.$$

This yields

$$m \leq \frac{-\varepsilon - \varepsilon^2}{-\frac{\varepsilon}{2}} m^* - \frac{\ln n}{-\frac{\varepsilon}{2}} = 2(1+\varepsilon)m^* + \frac{2\ln n}{\varepsilon}.$$

This follows from the following inequalities, which will also be useful later on. Elementary proofs are omitted

Lemma 4. *For any $\varepsilon \in [-\frac{1}{2}, \frac{1}{2}]$ we have*

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

□

4 Randomized Algorithms

We can obtain better bounds by turning to randomized algorithms. Before we do that, it is neccessary to specify the details of our model. For every possible *scenario*, i.e. a sequence of expert recommendations and actual results, we will bound the expected number of mistakes of our algorithm. Another way to look at this is that while we are still looking at the worst-case behaviour against an adversary that know our algorithm, we assume that he does not know the results of our algorithm's coin-flips. This kind of adversary is usually called an *oblivious* adversary in on-line algorithms.

Let us start by analyzing the randomized version of the halving algorithm.

Algorithm 4: The Randomized Halving Algorithm

```
Put all experts in a set S of trustworthy experts;
foreach round t do
    Follow a random expert from S;
    Remove from S all experts wrong in round t;
```

Theorem 5. *If there is a perfect expert, the Randomized Halving Algorithm makes at most* $\ln n$ *mistakes in expectation.*

Proof. Fix a scenario, i.e. sequence of expert recommendations and actual results. This also fixes the contents of S at all times. Let F^t be the fraction of the elements of S that are wrong in round t. Then F^t is also the probability of our algorithm making a mistake in round t. Therefore

$$E[m] = \sum_t F^t.$$

On the other hand, we have

$$|S^T| = n \prod_t (1 - F^t),$$

where S^t is the value of S after t rounds. After taking logarithms of both sides we get

$$\ln |S^T| = \ln n + \sum_t \ln(1 - F^t) \leq \ln n - \sum_t F^t.$$

Therefore

$$E[m] \leq \ln n - \ln |S^T| \leq \ln n - \ln 1 \leq \ln n.$$

□

Note that this is already a significant improvement over the deterministic version since the base of the logarithm has changed from 2 to e.

We can gain even more by applying randomization to the Weighted Majority Algorithm.

Algorithm 5: The Randomized Weighted Majority Algorithm

```
Set w_i := 1 for all experts i;
foreach round t do
    Follow an expert sampled proportionally to w_i;
    foreach expert i that turned out wrong do
        w_i := w_i(1 − ε);
```

Theorem 6. *For any* $0 < \varepsilon \leq \frac{1}{2}$*, the Randomized Weighted Majority Algorithm makes at most* $(1 + \varepsilon)m^* + \frac{\ln n}{\varepsilon}$ *mistakes in expectation.*

Proof. The proof is very similar to the previous one. For a fixed scenario, we define F^t to be the fraction of the total weight in round t corresponding to experts that turn out to be wrong. Then

$$E[m] = \sum_t F^t.$$

On the other hand, we have

$$W^T = n \prod_t (1 - \varepsilon F^t),$$

After taking logarithms of both sides we get

$$\ln W^T = \ln n + \sum_t \ln(1 - \varepsilon F^t) \leq \ln n - \sum_t \varepsilon F^t.$$

Therefore

$$E[m] \leq \frac{\ln n - \ln W^T}{\varepsilon}.$$

By looking at the best expert, we get

$$W^T \geq (1-\varepsilon)^{m^*},$$

and so

$$\ln W^T \geq m^* \ln(1-\varepsilon) \geq (-\varepsilon - \varepsilon^2) m^*,$$

and the claim follows. □

5 The General Setting

To unleash the full power of multiplicative weights updates, we introduce a couple generalizations/modifications:

- Instead of just being wrong or right, we introduce a continuum of possible outcomes. At the end of round t, every expert i receives a penalty $l_i^t \in [-\rho, \rho]$ (negative penalty means reward).
- Since in the previous step, we effectively removed the choices recommended by experts from the model (there are no heads or tails anymore), our algorithm now has to actually point the expert that it is going to follows instead of pointing the guessed outcome. The algorithm receives a penalty equal to the penalty of the chosen expert.
- We allow the algorithm to pick not just a single expert, but a probability distribution on the set of experts. If the algorithm picks distribution p^t in round t, then its penalty for this round is equal to $p^t l^t = \sum_i p_i^t l_i^t$ (assuming for simplicity that p^t is a row vector).

The last change is the most interesting one. Note that if we have $l_i^t \in \{0, 1\}$, then the previous two algorithms and their analysis still works. However, we do not use randomization anymore. We simply return the distribution used to sample the expert chosen to follow. To some extent the last change allows us to dispense with randomization altogether. One might say that we are now using potential randomness and not actual randomness.

Another interesting feature of the last change is that we can treat p^t as a convex combination of experts (and not a probability distribution, that is supposed to be sampled from). In many applications this combination is a very concrete object that has meaning beyond that of a convex combination. We will see two such applications in this lecture.

Let us now formulate the most general and final version of our algorithm.

Algorithm 6: The Multiplicative Weights Update Algorithm

```
Set w_i^0 := 1 for all experts i;
foreach round t do
    Set p_i^t = w_i^{t-1} / W^{t-1}, where W^{t-1} = Σ_i w_i^{t-1};
    Follow p^t;
    foreach expert i do
        w_i^t := w_i^{t-1} (1 − ε l_i^t / ρ);
```

Let l_{MWU} be the total loss of the above algorithm, i.e.

$$l_{MWU} = \sum_t p^t l^t = \sum_t \sum_i p_i^t l_i^t.$$

We then have

Theorem 7. *For any $0 < \varepsilon \leq \frac{1}{2}$ and any expert i*

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

Proof. We have

$$W^t = \sum_i w_i^t = \sum_i w_i^{t-1}(1-\varepsilon l_i^t/\rho) = W^{t-1} - \frac{\varepsilon}{\rho}\sum_i w^{t-1} l_i^t = W^{t-1} - \frac{\varepsilon}{\rho} W^{t-1} p^t l^t = W^{t-1}\left(1 - \frac{\varepsilon}{\rho} p^t l^t\right).$$

Therefore

$$\ln W^T = \ln\left(n \prod_t \left(1 - \frac{\varepsilon}{\rho} p^t l^t\right)\right) = \ln n + \sum_t \ln\left(1 - \frac{\varepsilon}{\rho} p^t l^t\right) \leq \ln n - \frac{\varepsilon}{\rho}\sum_t p^t l^t = \ln n - \frac{\varepsilon}{\rho} l_{MWU}.$$

But of course

$$\ln W^T \geq \ln w_i^T = \ln \prod_t \left(1 - \frac{\varepsilon}{\rho} l_i^t\right) = \sum_t \ln\left(1 - \frac{\varepsilon}{\rho} l_i^t\right) \geq -\frac{\varepsilon}{\rho}\sum_t \left(l_i^t + (l_i^t)^2 \frac{\varepsilon}{\rho}\right).$$

for every expert i. Joining these two bounds we get

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

□

Theorem 8. *For any $\delta > 0$, if we run the MWU algorithm with $\varepsilon = \frac{\delta}{2\rho}$ for $T \geq \frac{4\rho^2 \ln n}{\delta^2}$ rounds, then for any expert i*

$$\frac{l_{MWU}}{T} \leq \frac{\sum_t l_i^t}{T} + \delta.$$

Proof. Our proof strategy is very simple. We use the previous theorem and try to choose ε and T so that both $\frac{\varepsilon}{T}\sum_t |l_i^t|$ and $\frac{\rho \ln n}{\varepsilon T}$ are bounded by $\frac{\delta}{2}$. For the first of these inequalities it is enough to take $\varepsilon = \frac{\delta}{2\rho}$. For the second one, we need

$$\frac{\rho \ln n}{\varepsilon T} \leq \frac{\delta}{2},$$

i.e.

$$T \geq \frac{2\rho \ln n}{\varepsilon \delta} = \frac{4\rho^2 \ln n}{\delta^2}.$$

□

This last form of the error bound is, as we shall soon see, very useful in applications.

6 Proof of the Min-Max Theorem

We will now give an algorithmic proof of the famous Min-Max Theorem of von Neumann. This theorem concerns two-player zero-sum games. Each such game can be described by a payoff matrix A. The rows of this matrix correspond to the moves (pure strategies) available to the first player, the columns to the moves (pure strategies) available to the second player, and entry A_{rc} is the reward the first player gets if he plays r and his opponent plays c. Of course, for zero-sum games, the second player in this case loses A_{rc}. We will call the first player *the row player* and denote him with R, and we will call the second player *the column player* and denote him with C. If $A \in \mathbb{R}_{n\times m}$ (i.e. R has n moves available, and C has m moves), then any distribution x on $\{1,\ldots,n\}$ describes a *mixed strategy* for R, and any distribution on $\{1,\ldots,m\}$ a *mixed strategy* for C. The expected outcome of the game (for R) if R plays x and C plays y is $x^T Ay$.

Theorem 9 (von Neumann)**.**

$$\max_x \min_y x^T Ay = \min_y \max_x x^T Ay.$$

What this theorem says is the following. Suppose R chooses his mixed strategy x first and fixes it. Next C chooses a strategy y which fares best against x. How well can R do? The answer is

$$V_R = \max_x \min_y x^T Ay.$$

Note that we can assume here that C uses a pure strategy, and so y is a unit vector.

Similarly if we let C choose first, we get

$$V_C = \min_y \max_x x^T Ay.$$

Here, we can assume that x is a pure strategy.

Since R tries to maximize the outcome, and being force to choose a strategy first puts a player at a disadvantage, we always have $V_R \leq V_C$. What is not clear however is that the reverse inequality holds.

Proof (of von Neumann's Theorem). We will assume that entries of A all lie in $[-1,1]$. This can easily be guaranteed by scaling and shifting the values of A.

We will now let our MWU algorithm play the role of R against a player C which always picks an optimal pure counterstrategy to the strategy R plays. Here are the details:

- We set up an expert for each pure strategy $r \in R$. This expert always recommends playing r.
- In each round t our algorithm computes the convex combination p^t of pure strategies, interprets it as a mixed strategy, and plays it.
- C responds by playing the optimal pure response j^t to p^t.
- We set penalties of experts accordingly, i.e. $l_i^t = -A_{ij^t}$ (the minus sign is due to the fact that MWU framework uses penalties and A describes winnings).

We will show, that if the MWU algorithm uses ε small enough and the game above is played long enough, then in a certain sense R and C are using mixed strategies close to optimal strategies. In particular we will be able to show $V_R \geq V_C - \delta$ for any $\delta > 0$.

Note that in our case $\rho = 1$. Fix a $\delta > 0$ and let $T \geq \frac{4 \ln n}{\delta^2}$. Run the above game for T rounds with $\varepsilon = \frac{\delta}{2}$. From Theorem 8 we get

$$\frac{\sum_t p^t l^t}{T} \leq \frac{\min_i \sum_t l_i^t}{T} + \delta.$$

Let us first take a look at the L.H.S. of this inequality.

$$\sum_t p^t l^t = -\sum_t p^t A e_{j^t}$$

where e_{j^t} is a unit vector with a 1 at j^t-th position. Note that each expression of the form $p^t A e_{j^t}$ corresponds to a situation where R plays a strategy and C responds with the optimal pure counterstrategy. Therefore

$$\frac{\sum_t p^t l^t}{T} \geq -V_R.$$

As for the R.H.S., using a similar reasoning we have

$$\frac{\min_i \sum_t l_i^t}{T} = -\frac{\max_i \sum_t (e_i)^T A e_{j^t}}{T} = -\max_i (e_i)^T A \left(\frac{\sum_t e_{j^t}}{T} \right).$$

One can interpret $\frac{\sum_t e_{j^t}}{T}$ as the (empirical) strategy that C uses over all the rounds. Since we can choose i to be R's best pure response to this strategy, we get

$$\frac{\min_i \sum_t l_i^t}{T} \leq -V_C.$$

In the end, we get

$$-V_R \leq -V_C + \delta,$$

for arbitrarily chosen $\delta > 0$, which ends the proof. □

Remark 10. Note that we can actually extract R's and C's approximate optimal strategies from the computation which the MWU algorithm performs. To get one for R, we simply look the the round with the smallest penalty $p^t l^t$ and pick the corresponding p^t as our strategy. For C, we use C's empirical strategy over the whole run. (Prove that these strategies really are approximately optimal!)

7 Solving LPs

We will now use MWU algorithm to approximately solve LPs: we will find an almost feasible solution that has almost optimal objective function value. The exact meaning of this will become clear shortly.

Note that the standard approach to finding optimum strategies for two-person zero-sum games is based on solving LPs. In this sence the results of this section generalize those of the previous one. However, the constructions are significantly different and there is no direct correspondence between them.

Simplifying the problem. Our goal is to solve LPs of the form

$$\begin{aligned} \text{minimize } & c^T x \\ Ax &\geq b \\ x &\geq 0 \end{aligned} \tag{1}$$

To simplify the problem we will only design an algorithm that approximately answers questions of the form: For a given objective function value V, does there exist a feasible $\tilde{x}$ such that $c^T\tilde{x} = V$? Our algorithm will either (correctly) answer that no such $\tilde{x}$ exists or return $\tilde{x}$, s.t.

$$\begin{aligned} c^T\tilde{x} &= V \\ A\tilde{x} &\geq b - \varepsilon\mathbf{1} \\ \tilde{x} &\geq 0 \end{aligned} \tag{2}$$

By using binary search, we can use such an algorithm to find a solution arbitrarily close to optimum, and approximately feasible in a sense described above.

The primitive solver. Our algorithm is going to solve the above simplified problem by making several calls to a primitive solver that solves the problem for the case where $Ax \geq b$ is a single inequality.

Exercise 1. Show that if $Ax \geq b$ is a single inequality, then one can decide (exactly) whether there exists a feasible solution with a given objective function value. The time complexity should be polynomial in the number of variables, but in fact a linear solution is possible.

We will henceforth assume, that we have an oracle that given $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}$ and $V \in \mathbb{R}$ returns $x \geq 0$, such that $\alpha^T x \geq \beta$ and $c^T x = V$, or (correctly) answers that no such x exists.

The algorithm. We will again use the MWU algorithm. This time each expert will correspond to a single inequality. His job is to convince us that "this inequality is the most important one". In each round, we will compute p^t using the MWU algorithm and we will use the oracle with the inequality

$$(p^t)^T Ax \geq p^t b,$$

i.e. we combine the inequalities into a single inequality, using p^t as weights. If the oracle tells us that no feasible solution exists, then clearly we can answer "NO" and abort mission. Otherwise, we get a point x^t, which is feasible for combined inequality. How should we set up the penalties for experts? If x^t has a lot of slack in some inequality, it makes sense to penalize the corresponding expert – we are likely giving him a weight that is too large. This intuition leads to the following expression $l_i^t = a_i x^t - b_i$ (i.e. l_i^t is the slack of the i-th inequality for x^t).

If the algorithm is still running after a large enough number of rounds T, we return

$$\tilde{x} = \frac{1}{T}\sum_t x^t.$$

The analysis. We now need to figure out how many rounds we need to run the algorithm and how good is the solution we obtain. Assume for a moment that we know the maximum possible absolute value ρ of penalties that can occur in our algorithm (if it is less than 1 round it up to 1, to avoid problems if ρ is very small).

We can then set $\varepsilon = \frac{\delta}{2\rho}$ and $T = \frac{4\rho^2 \ln n}{\varepsilon^2}$ to get

$$\frac{1}{T}\sum_t p^t l^t \leq \frac{1}{T}\sum_t l_i^t + \varepsilon$$

for every expert i (i.e. for every LP inequality).

Looking at the L.H.S. we get

$$\sum_t p^t l^t = \sum_t \sum_i p_i^t (a_i x^t - b_i) \geq 0,$$

since x^t is a feasible solution of the combined inequality $(p^t)^T Ax \geq p^t b$.

On the other hand, the R.H.S. has a very simple interpretation

$$\frac{1}{T}\sum_t l_i^t = \frac{1}{T}\sum_t (a_i x^t - b_i) = a_i \tilde{x} - b_i.$$

Joining these two derivations we get

$$a_i \tilde{x} \geq b_i - \varepsilon,$$

which is exactly what we want.

There remains the problem of what the value of ρ is, since this value heavily influences the running time of the algorithm. Unfortunately we cannot discuss this issue here, due to lack of space. Let us remark that it depends on the shape of the LP polytope, and that for general LPs the method described here might be ineffective. However, there exist improved algorithms that are able to handle LPs with large ρ.

Let us now make a couple final remarks:

1. One might be worried that we only get an approximately feasible solution. However very often this is not a problem at all, if scaling the solution up yields a feasible solution with slightly increased objective function value. This is the case for many LPs, for example LPs for SET-COVER or VERTEX-COVER and other covering problems.

2. We do not need a perfect oracle. Even an approximate one (i.e. one that can return a solution that is only approximately feasible) will do, at the cost of slightly increasing the infeasibility of the solution.

3. The technique used here is in fact very general and it can solve problems of the form

$$\begin{aligned} &\text{minimize } g(x) \\ &f_i(x) \geq 0 \quad \forall_i \\ &x \geq 0 \end{aligned} \tag{3}$$

provided that:

1. we can solve the separation problem, i.e. decide (at least approximately) whether for a given V there exists $x \geq 0$ with $g(x) = V$ and $\sum p_i^t f_i(x) \geq 0$.

2. the functions f_i are concave, which is necessary to get

$$0 \leq \frac{1}{T} \sum_t l_i^t = \frac{1}{T} \sum_t f_i(x^t) \leq f_i(\tilde{x}).$$