CS787: Advanced Algorithms	**Scribe:** Evangelia Gergatsouli
Lecture 18: Online Learning/Prediction	**Date:** 05 April 2019

18.1 Setting and notation

Recall our setting from the previous lecture; there are n experts who make a prediction everyday and for each one of the T days:

1. Algorithm picks an expert $i_t \in [n]$ to follow
2. The adversary reveals the cost vector c_i at each time step t ($c_{i_t} \in [0, 1]$ for every expert i)
3. Algorithm incurs cost $c_t = c_{i_t t}$ at step $t \in [T]$

We will use the following notation for the algorithms and proofs that follow

- $C = \sum_{t=1}^{T} c_t$ is the total cost incurred by the algorithm after time T
- $C_i = \sum_{t=1}^{T} c_{i_t}$ is the cost of expert i for all time steps until T
- $c^* = \min_i c_i$ is the cost of the best expert

18.2 Previous Results

Previously we described three algorithms with the following competitive ratios

- *Majority*: if $c^* = 0$ (always-correct expert exists) then $c \leq \log_2 n$ (where $c_{i_t} \in \{0, 1\}$)
- *Weighted Majority*: $c \leq \frac{2}{1-\epsilon} c^* + \frac{2}{\epsilon} \log_2 n$ (where $c_{i_t} \in \{0, 1\}$)
- *Hedge*: $c \leq \frac{1}{1-\epsilon} c^* + \frac{1}{\epsilon} \log_2 n$ Regret of *Hedge* $= c - c^* = o(\epsilon T + \log n)$

Where ϵ is a parameter given to the algorithms above. Also, note that in all these cases we compare against the cost of the best expert and not the offline optimal, which maybe is an unreasonably hard benchmark to beat.

Now let's think what would happen if we just predicted who is the expert with least mistakes. This leads us to the following, quite natural algorithm.

18.3 Follow the Leader (FTL)

This algorithm essentially keeps track of a "leaderboard" for the experts. In every round, we know who is the expert that has made the minimum number of mistakes in the previous rounds. Then we just choose to follow whatever that expert says.

We denote by $c_i^t = \sum_{t'=1}^{t} c_{i'_t}$ to be the cost of expert i until time t. The procedure described before is called Follow The Leader (algorithm 1).

Algorithm 1: Follow The Leader

```
   Data: Before making decision at $t$, we know vector $c_i$ until $t-1$
1  foreach time $t \in [T]$ do
2      $i_t = \text{argmin}_i c_i^{t-1}$
3      Add $c_{i_t}$ to total cost
4  end
```

18.3.1 Worst Case Analysis

How bad can this algorithm actually go? Think that a bad input for this algorithm would be one where there are some experts are very close in the leaderboard and we keep changing the expert on the wrong time. Specifically, suppose there are only two experts, that always differ by 1 in the mistakes they have made, i.e. in every round one is correct and the other is wrong, and they keep alternating. So when we think that expert 1 would be correct, it's the round that this expert makes a mistake and expert 2 becomes the best. Then in the next round expert 2 makes a mistake so 1 becomes the best, and so on. Essentially, our algorithm is always one-step behind the optimal. This example is described in table 1.

Round	**Expert 1**	**Expert 2**
1	1/2	1
2	1	0
3	0	1
4	1	0
...	...	...
Final cost of each expert	$T/2$	$T/2$

Table 1: Bad example for FTL

The cost of the best expert is $T/2$, while FTL will pay T, so we pay twice as much as the best expert[1].

In this case, if we could just look only one step ahead, we would actually beat the best expert (we would make 0 mistakes, while the best expert makes $T/2$). This idea leads us to the next algorithm.

[1]This can actually be made worse with n experts

18.4 Be the Leader (BTL)

This is essentially the same algorithm as before, with the only difference being that when we try to decide for step t, we get to see the cost vector c_i in this step t. This procedure is described in algorithm 2.

```
Algorithm 2: Be The Leader
   Data: Before making decision at $t$, we know vector $c_i$ until $t$
1  foreach time $t \in [T]$ do
2      $i_t = \text{argmin}_i c_i^t$
3      Add $c_{i_t}$ to total cost
4  end
```

Note that this is not an algorithm that we can implement, since it requires that we have some information from the future. Instead, we use BTL to analyze the other algorithms. Also note that this is not the best algorithm if we were given the extra future information. Basically we want to use this extra bit information without making to much change to our original algorithm.

We show below that BTL actually outperformes the best expert, every time, so it makes sense for us to design an algorithm that tries to be close to BTL[2].

$$\begin{aligned}
c^* = C_{\widetilde{i}_T} = \sum_{t=1}^{T} c_{\widetilde{i}_T t} & & \text{total cost} = \text{cost of expert chosen at step } T \\
= c_{\widetilde{i}_T T} + C_{\widetilde{i}_T}^{T-1} & & \\
\geq c_{\widetilde{i}_T T} + C_{\widetilde{i}_{T-1}}^{T-1} & & \text{in step } (T-1) \text{ expert } \widetilde{i}_{T-1} \text{ is best} \\
= c_{\widetilde{i}_T T} + c_{\widetilde{i}_{T-1}(T-1)} + C_{\widetilde{i}_{T-1}}^{T-2} & & \\
\geq c_{\widetilde{i}_T T} + c_{\widetilde{i}_{T-1}(T-1)} + C_{\widetilde{i}_{T-2}}^{T-2} & & \\
\ldots \quad \ldots & & \\
\geq \sum_{t=1}^{T} c_{\widetilde{i}_t t} = BTL & &
\end{aligned}$$

Observe that BTL has zero regret, while the regret of FTL is essentially the number of times the best leader on leaderboard changes. In makes sense then to try and modify FTL to make changes in the leader more difficult to happen, maybe using randomness. This idea leads us to the next algorithm.

[2]Note also that we always analyze algorithms in the adversarial setting ; there is an adversary who tries to give us the worst possible input. Another setting is the stochastic setting, where we know some distribution for the input. The adversarial setting is in some sense the "worst case analysis"

18.5 Follow the Perturbed Leader (FTPL)

In this algorithm, we make use of randomness by introducing one random variable X_i for every expert, where each variable is exponentially distributed with parameter ϵ, i.e. $X_i \sim \exp(\epsilon)$. We can think that, in order to choose the X_i's, for every expert i we independently flip a coin with bias ϵ [3], and then taking X_i to be the total number of flips.

Algorithm 4: Follow The Perturbed Leader

```
  Data: Before making decision at $t$, we know vector $c_i$ until $t-1$, $X_i \sim exp(\epsilon)$
1 foreach time $t \in [T]$ do
2     $i_t = \text{argmin}_i(c_i^{t-1} - X_i)$
3     Add $c_{i_t}$ to total cost
4 end
```

18.6 Be the Perturbed Leader (BTPL)

This is equivalent to BTL, the only difference from FTPL is that we know the cost vector until the round we are now in.

Algorithm 6: Be The Perturbed Leader

```
  Data: Before making decision at $t$, we know vector $c_i$ until $t$, $X_i \sim exp(\epsilon)$
1 foreach time $t \in [T]$ do
2     $i_t = \text{argmin}_i(c_i^{t} - X_i)$
3     Add $c_{i_t}$ to total cost
4 end
```

We denote by R_{alg} the regret of algorithm *alg*. We will bound the regret of FTPL using the following claims.

Claim 18.6.1 *Cost of BTPL* $\leq c^* + \max_i X_i$

Proof: We prove this in a way similar to the proof for BTL. Consider the cost of the best expert c^*, we get

[3] by bias we mean the probability of the coin coming up heads

$$
\begin{aligned}
c^* &\geq \sum_{t=1}^{T} c_{\tilde{i}_T t} - X_{\tilde{i}_T} + X_{i^*} && \text{since } c^T_{\tilde{i}_T} - X_{\tilde{i}_T} \leq c^T_i - X_i \text{ for step } T \\
&= c_{\tilde{i}_T T} + C^{T-1}_{\tilde{i}_T} - X_{\tilde{i}_T} + X_{i^*} \\
&\geq c_{\tilde{i}_T T} + C^{T-1}_{\tilde{i}_{T-1}} - X_{\tilde{i}_{T-1}} + X_{\tilde{i}_T} + X_{i^*} && \text{since } c^{T-1}_{\tilde{i}_{T-1}} - X_{\tilde{i}_{T-1}} \leq c^{T-1}_{\tilde{i}_T} - X_{\tilde{i}_T} \text{ for step } T-1 \\
&\ldots && \ldots \\
&\geq \sum_{t=1}^{T} c_{\tilde{i}_t t} - X_{\tilde{i}_1} + X_{i^*}
\end{aligned}
$$

■

Claim 18.6.2 *The regret of FTPL is less than the regret of BTPL plus the expected number of steps where* $i_t \neq \tilde{i}_T$

$$R_{FTPL} \leq R_{BTPL} + \mathbf{E}\left[\mathbb{1}_{\{step\ i: i_t \neq \tilde{i}_t\}}\right]$$

Claim 18.6.3 *At any time t, the probability we change the leader is small.* $\boldsymbol{Pr}[i_t \neq \tilde{i}_t] \leq \epsilon$

Proof: We will prove the equivalent statement of this claim, namely that

$$\mathbf{Pr}\left[c^{t-1}_{i_t} - X_{i_t} \leq c^{t-1}_i - X_{i-1},\ \text{for all } i \neq i_t\right] \geq 1 - \epsilon$$

where randomness in the above expression is over the variables X_i. Assume now that we have fixed all costs up to step $t-1$. In order to find the next costs we need to flip the coin X_i for every expert i, and reduce the value of c^{t-1}_i by 1 every time the coin does not come up heads. We always flips the coins for the highest cost expert. At some point, only 2 experts are left that are still flipping coins, if one of them finds heads and stop, and now the other one flips the coin until stop, the probability that we take one more step is $1-\epsilon$ (i.e.if we don't find heads again). That means, that the probability that the gap between them is ≥ 1, is at least $1-\epsilon$. You can see this in figure 18.6.1 below ; the continuous lines are the initial costs c^{t-1}_i, while the dotted lines below each of them are for coin flips with tails and the dotted line with star are when coin flips come up head and that's the cost for that expert.

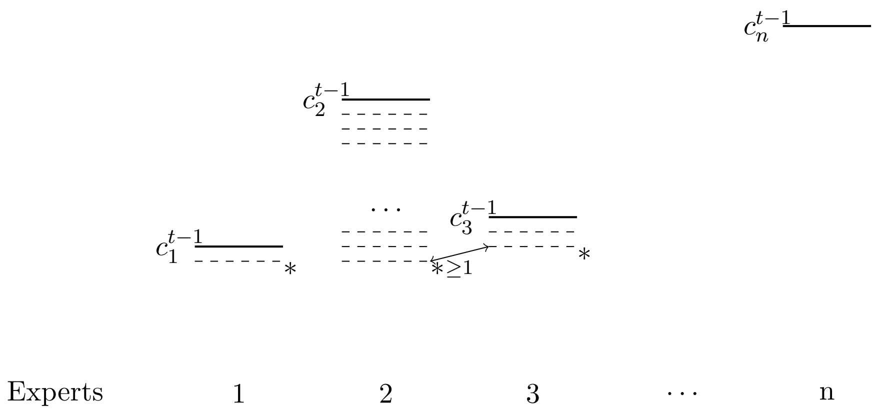

Figure 18.6.1: Proof of claim 18.6.3

■

The intuition for using randomness in this algorithm is to decrease the value of each expert is that if there are many experts that are better, they will eventually dominate the leaderboard. These experts are all good enough, so the X_i's ensure that we will not switch much among them. We use flipping coins to add adequate amount of pertubation with enough randomness instead of maintain of a random probability distribution. This way can be beneficial under certain circumstances. For example, if we want to know the best path between two locations without knowing costs of each path everyday. There are exponentially many paths and edges. It's easier to maintain a leaderboard as oppose to maintain an explicit distribution.

Claim 18.6.4 $E[\max_i X_i] \leq O(\log n/\epsilon)$

Proof: Left as an exercise. ■

Finally, putting all of the claims together we get

$$\begin{aligned}
R_{FTPL} &\leq R_{BTPL} + \mathbf{E}\Big[\mathbb{1}_{\{\text{step } i: i_t \neq \tilde{i}_t\}}\Big] && \text{from claim 18.6.2} \\
&\leq R_{BTPL} + \epsilon T && \text{from claim 18.6.3} \\
&\leq E[\max_i X_i] + \epsilon T && \text{from claim 18.6.1} \\
&\leq O(\log n) + \epsilon T && \text{from claim 18.6.4}
\end{aligned}$$

18.7 Multi-armed Bandits

In this case we have n experts/arms, and we need to decide which one to choose in each of the T rounds. The setting is very similar to before:

1. Algorithm picks an arm $i \in [n]$ to pull

2. The adversary reveals the cost c_{i_t} (the cost of the chosen arm only) at each time step t
3. Algorithm incurs cost $c_t = c_{i_t t}$ at step $t \in [T]$

The only difference is that we don't get to see the cost of the other experts, other than the one we picked. Sometimes it will be good to pull a random arm to explore the cost for unknown arms; sometimes it makes sense to pick the best expert from BTPL to exploit the information we collected. This is called an *Explore/exploit* strategy; balance between how much information you try to find (explore) and how much the actual cost is (exploit).