CS 170 Section 13

Multiplicative Updates

Owen Jow

April 25, 2018

University of California, Berkeley

Table of Contents

Multiplicative Updates Intro

The Experts Problem

- Every day,
  you enter a transaction in which you lose between 0 and 1 dollars
  - Life is hard
- There are n experts, each of whom gives different advice
- Instead of making your own decisions, you choose an expert every day and follow his advice
- The next day you find out how all the experts performed, and you can choose another expert if you wish
- **Goal:** minimize regret

Terminology

- There are n experts
- There are T days (T is very large)
- The ith expert on day t costs you $c_i^t \in [0, 1]$
- You choose expert $i(t)$ on day t
- R is your regret

Regret

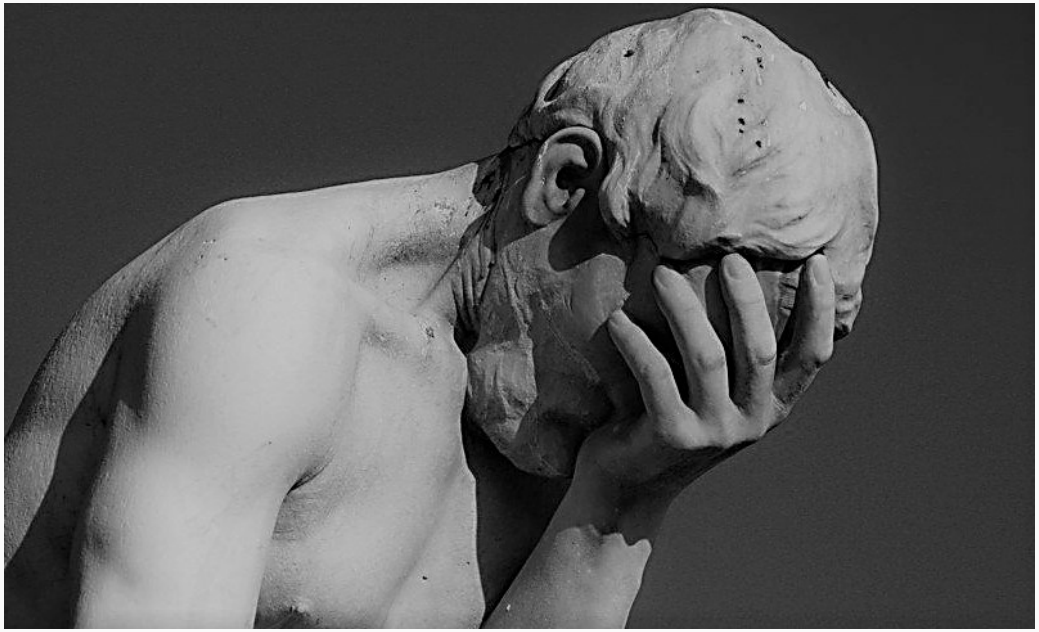

Figure 1: we would like to minimize our **regret** R.

$$R = \frac{1}{T}\left(\sum_{t=1}^{T} c_{i(t)}^{t} - \min_{i} \sum_{t=1}^{T} c_{i}^{t}\right)$$

i.e. on average ((how you did) − (how the best expert did))

Goal Reframed

- More specifically, you would like an algorithm for choosing experts with the result that $R \approx 0$ no matter what c_i^ts the environment throws at you (i.e. even in the worst case)
- For this you can use **multiplicative weight updates**

Notes

- You want your algorithm to do as well as the one that picks the best expert from the start and sticks with him
- Regret is defined at the end (how did you do in comparison to how you'd have done if you chose the best expert at the start and followed him every day?)
- It is impossible to match the best expert on a day-to-day basis, but it is possible to match the single best expert throughout
- The adversary is the environment, which provides the cost values

Multiplicative Weight Updates

MWU is a randomized algorithm.

It chooses expert i on day t with weight $w_i^t > 0$.

Algorithm 1 Multiplicative Weight Updates

1: Initialize all weights to $w_i^0 = 1$.
2: **for** $i = 1$ to T **do**
3: Choose expert i with probability $\frac{w_i}{\sum_j w_j}$
4: Update weights for all experts: $w_i^{t+1} = w_i^t \cdot (1 - \epsilon)^{c_i^t}$
5: **end for**

Multiplicative Weight Updates

- $(1-\epsilon)^{c_i^t}$ will be less than or equal to 1. It'll be much less than 1 if the expert ruined you; the bigger c_i^t is, the more you punish expert i.
  - In the words of a certain theoretical computer scientist, "c_i^T is the amount of money this bastard made you pay."
- Weights "absorb" all past performances of experts
- Experts who perform the best end up with the highest weights

Multiplicative Weight Updates

- This algorithm can be proven to give almost zero regret.
- The proof is left as an exercise.
- Just kidding. For the proof, see the **notes**.

$$
\begin{aligned}
R = \frac{1}{T}(\mathrm{MWU} - \mathrm{OPT}) &\leq \frac{\ln n}{\epsilon T} + \epsilon \frac{\mathrm{OPT}}{T} \\
&\leq \frac{\ln n}{\epsilon T} + \epsilon \\
&\leq 2\sqrt{\frac{\ln n}{T}}
\end{aligned}
$$

Notes

- With this algorithm, higher T means smaller regret.
- MWU punishes bad experts exponentially severely. By the crushing weight of exponentiation, if an expert is the best you'll be choosing him all the time.

Life Advice

If you want zero regret in life, notice what works in a very conservative fashion – by giving it a little more weight every time. In the long run, this means perfection.

A theoretical computer scientist

Follow the Regularized Leader

Exercise 1a

- You are playing T rounds of a game
- At round t you pick strategy $i \in \{1, ..., n\}$ and receive payoff $A(t, i) \in [0, 1]$
- What happens if you choose at each round the strategy which has given the highest average payoff so far? (*Even though you throw in your lot with one strategy, you get to observe how all of them do.*)

Exercise 1b

- The problem: if you choose strategies deterministically, an adversarial environment can design payoffs to ruin you
- So let's try a randomized strategy
  - To the adversary: good luck outplaying randomness
- Pick each strategy at random from a distribution D_t

Exercise 1b

- D_t assigns a probability $p_t(i)$ to each strategy i
- At round t, "follow the leader" will approximately maximize

$$\sum_{i=1}^{n} \left(p_t(i) \cdot \sum_{\tau \in \{1,\ldots,t-1\}} A(\tau, i) \right)$$

- Why is this no better than before?

Exercise 1c

- Let's add an entropy regularizer, now maximizing at time step t

$$\sum_{i=1}^{n} \left[\left(p_t(i) \cdot \sum_{\tau \in \{1,\ldots,t-1\}} A(\tau, i) \right) - \eta p_t(i) \ln p_t(i) \right]$$

- Suddenly, "follow the regularized leader" is the same as MWU.
- Show that for any distribution p_t, our objective is at most

$$\eta \ln \left(\sum_{i=1}^{n} e^{\sum_{\tau \in \{1,\ldots,t-1\}} \frac{A(\tau,i)}{\eta}} \right)$$

Exercise 1d

When computing p_t using multiplicative weight updates, we can say for some choice of ϵ (dependent on η) that the objective

$$\sum_{i=1}^{n} \left[\left(p_t(i) \cdot \sum_{\tau \in \{1,\ldots,t-1\}} A(\tau, i) \right) - \eta p_t(i) \ln p_t(i) \right]$$

is equal to

$$\eta \ln \left(\sum_{i=1}^{n} e^{\sum_{\tau \in \{1,\ldots,t-1\}} \frac{A(\tau,i)}{\eta}} \right)$$

Show this. Also, how does ϵ depend on η?