Lecture 16

The Multiplicative Weights Algorithm*

In the next couple lectures, we will devise modular, iterative algorithms for solving LPs and SDPs. "Multiplicative weights" is a retronym for the simple iterative rule underlying these algorithms; it is known by different names in the various fields where it was (re)discovered. Check out the survey by Arora, Hazan and Kale [AHK05]; our discussion will be based on their treatment. Due to its broad appeal, we will consider multiplicative weights in more generality than we need for solving LPs and SDPs. In this lecture, we'll introduce some strategies for playing a prediction game. We'll tweak the game to suit our optimization needs. Finally, we'll play the tweaked game with a strategy called Hedge.

16.1 Warmup: prediction with expert advice

The following sequential game is played between an omniscient Adversary and an Aggregator who is advised by N experts. Special cases of this game include predicting if it will rain tomorrow, or if the stock market will go up or down.

For $t = 1, \ldots, T$:

1. Each expert $i \in [N]$ advises either yes or no.
2. Aggregator predicts either yes or no.
3. Adversary, with knowledge of the expert advice and Aggregator's prediction, decides the yes/no outcome.
4. Aggregator observes the outcome and suffers if his prediction was incorrect.

Naturally, Aggregator wants to make as few mistakes as possible. Since the experts may be unhelpful and the outcomes may be capricious, Aggregator can hope only for a relative

*Lecturer: Anupam Gupta. Scribe: Shiva Kaul.

performance guarantee. In particular, Aggregator hopes to do as well as the best single expert in hindsight [1]. In order to do so, Aggregator must track which experts are helpful. We will consider a few tracking strategies. Almost every other aspect of the game - that advice is aggregated into a single value, that this value is binary, and even that the game is sequential - is not relevant; we will generalize or eliminate these aspects.

If there is a perfect expert, then an obvious strategy is to dismiss experts who aren't perfect. With the remaining experts, take a majority vote. Every time Aggregator makes a mistake, at least half of the remaining experts are dismissed, so Aggregator makes at most $\log_2 N$ mistakes. We can use the same strategy even when there isn't a perfect expert, if we restart after every expert has been eliminated. If the best expert has made M mistakes by time T, then Aggregator has restarted at most $M+1$ times, so it has made at most $(M+1)\log_2 N$ mistakes. This bound is rather poor since it depends multiplicatively on M.

16.1.1 Fewer mistakes with Weighted Majority

We may obtain an additive mistake bound by softening our strategy: instead of dismissing experts who erred, discount their advice. This leads to the Weighted Majority algorithm of Littlestone and Warmuth [LW89]. Assign each expert i a weight $w_i^{(1)}$ initialized to 1. Thereafter, for every t:

- Predict yes/no based on a weighted majority vote per $\vec{w}^{(t)} = (w_1^{(t)}, \ldots, w_N^{(t)})$
- After observing the outcome, for every mistaken expert i, set $w_i^{(t+1)} = w_i^{(t)}/2$

Theorem 16.1. *For any sequence of outcomes, duration T and expert i,*

$$\#\text{ of WM mistakes} \leq 2.41 \cdot (\#\text{ of } i\text{'s mistakes}) + \log_2 N \quad \text{* 2.41}$$

Proof. Let

$$\Phi^{(t)} = \sum_{i \in [N]} w_i^{(t)}$$

be a 'potential' function. Observe the following facts:

- By definition, $\Phi^{(1)} = N$
- Also by definition, $\frac{1}{2}^{\#\text{ of } i\text{'s mistakes}} \leq \Phi^{(T+1)}$
- At any τ when WM errs, at least half of the weight gets halved:

$$\Phi^{(\tau+1)} \leq \frac{3}{4}\Phi^{(\tau)}$$

This should be: # of WM mistakes

This implies

$$\Phi^{(T+1)} \leq \frac{3}{4}^{\#\text{ of } i\text{'s mistakes}} \cdot \Phi^{(1)}$$

[1] The excess number of mistakes is called (external) regret.

Combining these facts yields

$$\frac{1}{2}^{\#\text{ of } i\text{'s mistakes}} \leq \frac{3}{4}^{\#\text{ of WM mistakes}} \cdot N$$

Taking logarithms of both sides,

$$-(\#\text{ of } i\text{'s mistakes}) \leq \log_2 N + \log_2(3/4) \cdot \#\text{ of WM mistakes}$$

so finally

$$\#\text{ of WM mistakes} \leq (1/\log_2(4/3)) \cdot (\#\text{ of } i\text{'s mistakes}) + \log_2 N$$

The parenthesis should end after log_2 N

□

The unseemly leading constant is a consequence of our arbitrary choice to halve the weights. If we optimize ϵ in the update rule

$$w_i^{(t+1)} = w_i^{(t)}/(1+\epsilon)$$

then we may achieve

$$\#\text{ of WM mistakes} \leq 2(1+\epsilon) \cdot (\#\text{ of } i\text{'s mistakes}) + O(\log N/\epsilon).$$

Bibliography

[AHK05] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta algorithm and applications. Technical report, Princeton University, 2005. 16

[FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55:119–139, August 1997. 16.3

[Gup11] Anupam Gupta. Lecture #17: Multiplicative weights, discussion. http://lpsdp.wordpress.com/2011/11/09/lecture-17-multiplicative-weights-discussion/, 2011. 16.2

[LW89] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. In *FOCS*, pages 256–261, 1989. 16.1.1