Advanced Algorithms 2012A
Lecture 8 – Prediction using experts and fast LP solver*

Robert Krauthgamer

1 Prediction using experts

We describe a method of using advice given daily by n experts. These algorithms were devloped in the context of machine learning, specifically online decision making and regret minimization.

1.1 A perfect expert

We start with a simple setting to demonstrate the method. There is one stock which every day can go either up or down. An online learning algorithm sees, every day, the predictions of n experts $[n]$, makes it own decision (i.e. "act" in the morning), and then observes the outcome (see stock price in the evening).

Assume for now at least one expert is perfect–never makes a mistake. Then we can maintain a set E of all experts that made no mistake so far. But how to predict?

Algorithm 1:

Initialize $E \leftarrow [n]$. Then Daily:

1. Predict: according to majority of experts in E

2. Update: remove from E all experts who predicted incorrectly.

Proposition 1: If there is a perfect expert, this algorithm makes at most $\log n$ mistakes.

Proof: Whenever the algorithm makes a mistake, at least half the experts were wrong, hence $|E|$ decreases by factor 2.

Key features (1) The algorithm evaluates the experts' past performance (gives scores); and (2) Whenever the algorithm errs, the analysis gains valuable information (potential function)

Exer: Show that if the best expert make m^* mistakes (rather than perfect), then a variant of this algorithm makes at most $O(m^* \log n)$ mistakes. (Hint: what should the algorithm do when E

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

becomes empty?)

1.2 Weighted Majority

What should the algorithm do when experts might make mistakes? We penalize them by reducing their score (called weight), say by factor 2.

WM algorithm:

Initialize all $w_i = 1$.

1. Predict: according to weighted majority of experts

2. Update weights: if expert i predicted incorrectly set $w_i \leftarrow w_i/2$.

Proposition 2: If the best expert makes only m^* mistakes and the algorithm makes M mistakes, then

$$M \leq 2.41(m^* + \log n).$$

Proof: The proof, seen in class, uses $W = \sum_i w_i$ as a potential function (sort of a budget for future mistakes).

1.3 Weighted Majority with multiplicative factor $1 - \varepsilon$

Theorem 3: [Littlestone-Warmuth'89] The number of mistakes M made by WM algorithm with multiplicative factor $1 - \varepsilon \geq 1/2$ is

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

Proof: Was seen in class and is similar to before.

Exer: Prove that for every deterministic algorithm there is an input sequence where the algorithm makes $M \geq 2m^*$ mistakes.

This exercise shows that the leading factor 2 in the theorem is tight. We can do better using a randomized algorithm, i.e. probabilistic strategies.

Exer: Suppose the total number of days T is known. Then the number of mistakes can be bounded by $O(\sqrt{T \log n})$. Show also a lower bound of $\Omega(\sqrt{T})$ for two experts. Hint: Let the two experts have random but opposite predictions.

1.4 Multiplicative Weights

We generalize the setting: losses/costs are in $[0, 1]$, and the losses of experts are unrelated to each other. On the other hand, we allow a randomized algorithm. Let $m_i^{(t)} \in [0, 1]$ be the loss of expert i at time t.

MW Algorithm:

Initialize all $w_i^{(1)} = 1$. At each time $t = 1, \ldots, T$:

1. Predict: choose expert i with probability proportional to w_i^t.

2. Update weights: $w_i^{t+1} \leftarrow w_i^t(1 - \varepsilon m_i^t)$.

Remark: a different version of the update rule is $(1-\varepsilon)^{m_i^t}$. Another one is $e^{-\varepsilon m_i^t}$, called Hedge [Freund-Schapire'97]. We describe a version from [Arora-Hazan-Kale'05].

Theorem 4: Denote the expected loss of the MW algorithm by $\bar{M}$, and that of the best expert by $m^* = \min_i \sum_t m_i^t$, and let $\varepsilon \leq 1/2$. Then

$$\bar{M} \leq (1+\varepsilon)m^* + \frac{\ln n}{\varepsilon}.$$

Proof Was seen in class, using the potential function $W^t = \sum_i w_i^t$.

Remark [mixed experts/strategies]: In some contexts, it make sense to choose a probability vector (convex combination) over the experts, i.e. $\vec{p} \in \mathbb{R}^n_{\geq 0}$ with $\sum_i p_i = 1$. Then the MW algorithm need not be random, but rather use (deterministically) the distribution determined by the weights w_i, and have the same loss as $\bar{M}$ above. We can further compare the algorithm to any probability distribution $\vec{p}$ over experts, because then the loss $\sum_{t\in[T]} \sum_{i\in[n]} p_i m_i^t = \sum_{i\in[n]} p_i \sum_{t\in[T]} m_i^t$ is a weighted average of the losses of the different experts $i \in [n]$, hence minimized by choosing p to be some e_i (i.e. some expert $i \in [n]$).

Exer: Suppose the experts loss is in the range $[0, \rho]$. Show a similar algorithm with analogous bound $\bar{M} \leq (1+\varepsilon)m^* + \frac{\rho \ln n}{\varepsilon}$.

Exer: Suppose the experts have gains $g_i \in [0,1]$. Show that a similar algorithm with update $w_i(1+\varepsilon g_i^{(t)})$ achieves expected gain $\bar{G} \geq (1-\varepsilon)G^* - \frac{\ln n}{\varepsilon}$.

2 Fast approximate LP solver

2.1 Approximating feasibility LP via MW algorithm

We now use the MW technique to quickly find approximate solutions to LP. We focus on feasibility problem:

$$\exists ? x \in P : \; Ax \geq b$$

where $P \subset \mathbb{R}^n$ is convex, A is an $m \times n$ real matrix, and $x \in \mathbb{R}^n$. Intuitively, P represents the easy constraints (say nonnegatiivity) and A is the "actual" problem to solve.

We will design a (deterministic) meta-algorithm for solving this LP, that either solves the LP with additive approximation $\delta > 0$, i.e., output a solution x such that $A_i x \geq b_i - \delta$ for all $i \in [m]$, or declares failure and provides a certificate of infeasibility.

The idea: Use MW to reduce the problem to multiple easier instances with only a single constraint of the form:

$$\exists ? x \in P : \; p^{\perp} A x \geq p^{\perp} b,$$

where $\vec{p} \in \mathbb{R}^m_{\geq 0}$. This can be done for instance by maximizing $p^{\perp} A x$ over P, which is very easy if P is nonnegativity constraints.

The oracle: Assume henceforth there is a subroutine, called oracle, that given p, finds a solution x (for the p-combination of constraints) or reports that no such x exists. In the latter case, the original problem is not feasible a la Farkas Lemma, hence we can stop and declare failure with certificate p. We further assume there is $\rho > 0$ such that the oracle is ρ-bounded meaning that the reported solution satisfies $A_i x - b \in [-\rho, \rho]$. (This ρ is called the *width* of the LP.)

Theorem 5 (with cheating): Suppose the feasibility problem $\{x \in P : \; A_x \geq b\}$ has a ρ-bounded oracle, and let $0 < \delta < \rho$ be the desired approximation. Then feasible instances can be solved up to additive error δ using only $t^* = O(\frac{\rho^2 \ln m}{\delta^2})$ calls to the oracle in $O(m)$ time per call.

The algorithm: The solver applies the MW algorithm in the probability vector (deterministic) "mode" (from above Remark), with m experts (=constraints) and $\varepsilon \leq \frac{1}{2}$ to be chosen later as $\varepsilon = \frac{\delta}{2\rho}$. Let the vector p be the output of the MW algorithm (it is the weight vector w normalized to be a probability vector). Recall that initially p is the uniform distribution. At each iteration $t = 1, \ldots, t^*$ (for t^* chosen as in the theorem), the solver algorithm takes the p received from MW algorithm and feeds it to the oracle to obtain a feasible solution x^t satisfying ${p^{(t)}}^{\top} A x^{(t)} \geq {p^{(t)}}^{\top} b$.

The solver then then tells the MW algorithm that the experts' loss (cost) at iteration t is $m^{(t)} = \frac{1}{\rho}[A x^{(t)} - b]$. (Remark: I am cheating here, since Theorem 4 does not allow negative m_i, but it can be corrected using a more careful analysis of MW or via the Hedge algorithm.) At the end, the solver reports the average of the solutions found in the process $\bar{x} = \frac{1}{t^*} \sum_{t=1}^{t^*} x^{(t)} \in P$ (recall P is convex).

The analysis was seen in class.

2.2 Approximating Multicommodity flow

Recall the multicommodity problem can be formulated as LP:

$$\boxed{\begin{array}{lll} \text{maximize} & \sum_i \sum_{p \in P_i} f_p^i & \\ \text{subject to} & \displaystyle\sum_{i \in [k]} \sum_{p \in P_i : e \in p} f_p^i \leq c_e & \forall e \in E \\ & f_p^i \geq 0 & \forall i \in [k], \forall p \in P_i \end{array}} \tag{1}$$

Let $P' = \cup_i P_i$ be the set of all $s_i - t_i$ paths. Denote the value of the optimal flow by F^*, and suppose we know it (by binary search). Assume by normalization the minimum edge capacity is $c_{\min} = 1$.

Theorem 6: A multicommodity flow of value $(1-\delta)F^*$ can be computed in time $O(\delta^{-2}F^{*2}mk \ln m)$, where $m = |E|$.

Remark: A more careful argument can reduce the dependence on $\rho = F^*$ to be linear. With more work to "control" the width, the bound can be made independent of F^*.

We briefly discussed in class a sketch of the proof. It uses Theorem 5 to solve the feasibility problem

$$\exists ? \vec{f} \in P : \quad \forall e \in E, \sum_{p \in P' : e \in p} f_p / c_e \leq 1$$

over the polytope

$$P = \{\vec{f} : \ \forall p \in P', \ f_p \geq 0 \text{ and } \sum_{p \in P'} f_p = F^*\}.$$

The oracle computation turns out to be to compute shortest-path between k pairs and output the best one.