CS 683 — Learning, Games, and Electronic Markets **Spring 2007**

Notes from Week 8: Multi-Armed Bandit Problems

Instructor: Robert Kleinberg *12-16 Mar 2007*

1 The multi-armed bandit problem

The multi-armed bandit problem is like the best-expert problem studied earlier in this class, except that the algorithm gets limited feedback at the end of each experiment. Specifically, the only information the algorithm learns after picking a strategy x_t at time t is the cost (or payoff) $c_t(x_t)$ of the chosen strategy at that time; it does not learn the costs of other strategies. The name "multi-armed bandit problem" is derived from a gambling metaphor: slot machines are sometimes jokingly called "one-armed bandits," so a multi-armed bandit problem refers to a situation in which a gambler has access to n different slot machines and faces the problem of deciding which slot machine to play in every period so as to maximize the expected payoff.

Multi-armed bandit problems are an important abstraction for decision problems that incorporate an "exploration versus exploitation" trade-off. This trade-off can be seen clearly in the gambling scenario described above. After the gambler has discovered a slot machine whose average payoff is fairly good, there is a tension between continuing to play this slot machine (exploitation) versus trying other alternatives that have never been tested or that have only been tested infrequently (exploration). Because this type of trade-off is pervasive in the world of on-line decision problems, there are many applications of multi-armed bandit algorithms: some of these applications are described in Section 2.

1.1 Formal statement of the problem

A multi-armed bandit problem is specified by a set of strategies $\mathscr{S}$ — which will be identified with the set $[n]$ in all of the lectures this week — and a set Γ of possible *cost functions* on $\mathscr{S}$. In all of the lectures this week, Γ will be the set $[0,1]^{\mathscr{S}}$ of all functions from $\mathscr{S}$ to $[0,1]$.

Let $c_1, c_2, \ldots, c_T$ denote a sequence of cost functions chosen by an oblivious adversary. A *multi-armed bandit algorithm* is a randomized online algorithm for choosing a sequence of strategies $x_1, x_2, \ldots, x_T \in \mathscr{S}$ such that for all $t \geq 1$, the algorithm's choice of x_t depends only on its own random bits and on the values $c_1(x_1), c_2(x_2), \ldots, c_{t-1}(x_{t-1})$. Note the crucial difference between this problem and the best-expert problem, in which the choice of x_t is allowed to depend on the entire sequence of functions $c_1, \ldots, c_{t-1}$ and not only the sequence of numbers $c_1(x_1), \ldots, c_{t-1}(x_{t-1})$ obtained by evaluating the functions at the chosen strategies.

The algorithm's objective is to minimize the average cost of the strategies it chooses. Its normalized regret is defined by the formula:

$$\hat{R}(\mathsf{ALG}, T) = \max_{x \in \mathscr{S}} \frac{1}{T} \mathbf{E}\left[\sum_{t=1}^{T} c_t(x_t) - c_t(x)\right].$$

In the foregoing, we have assumed that the sequence of cost functions $c_1, c_2, \ldots, c_T$ is specified by an oblivious adversary. It is also possible to consider the multi-armed bandit problem with an adaptive adversary; in this case, c_t is allowed to depend on $x_1, x_2, \ldots, x_{t-1}$ but not on $x_t, x_{t+1}, \ldots, x_T$.

2 Applications of bandit algorithms

As stated above, bandit algorithms have many applications. Here are a few of them.

Design of ethical clinical trials. Suppose that you are evaluating n possible treatments for a disease, to determine which one is most effective. A standard way to do this is to design a clinical trial with a pool of T subjects partitioned randomly into n groups of equal size. Each treatment is administered to one group of subjects, and the results across different groups are compared to see which treatment is most effective.

When the experiment is being performed over time — so that some of the subjects arrive after the treatments have already been tested on earlier subjects — it is beneficial to adjust the rule for partitioning subjects into groups as time progresses. In later stages of the trial, a greater fraction of the subjects should be assigned to treatments which have performed well during the earlier stages of the trial. This increases the expected benefit of the treatments to the subjects in the clinical trial, and it also has the effect of increasing the amount of data collected for the most successful treatments.

In the extreme case in which a subject's response to a treatment can always be evaluated before the next subject arrives, this problem becomes a multi-armed bandit problem. Here $\mathscr{S}$ is the set of treatments, the time periods $1, 2, \ldots, T$ correspond to the sequence of subjects participating in the trial, and the cost $c_t(x_t)$ measures the disutility of applying treatment x_t to subject t. (A cost of 0 corresponds to a successful treatment; a cost of 1 corresponds to the worst possible outcome, e.g. the death of the subject.)

Server selection in networks. A recurring motif in networking is server selection: the process by which clients choose one server from among a set of servers supplying a specified service. For example, a DNS resolver looks up a hostname in a given domain by selecting one DNS server from among the list of authoritative name servers for the domain being queried. It is desirable to minimize the server's response time. (Some servers will consistently have a faster response time, e.g. because they are located closer to the client.) This is a multi-armed bandit problem in which $\mathscr{S}$ is the set of authoritative name servers, and c_t is the function which specifies the response time of each name server at time t.

Internet ad placement. Suppose you are launching an Internet advertising campaign for a new product. You have bought advertising space on a website, and each time a user visits the site you must choose to display one of n possible advertisements. The payoff (negative cost) of displaying an advertisement to a user is 1 if the user clicks on the advertisement, 0 otherwise. (Or perhaps it is better to define the payoff to be the amount of revenue gained from the user's possible purchase of your product, minus the cost charged to you because the user clicked on your advertisement.) In any case, this is a multi-armed bandit problem in which $\mathscr{S}$ is the set of advertisement which you might display, and $[T]$ is identified with the sequence of users visiting the website.

Pricing identical goods for sale. Suppose you are using a website to sell identical copies of a good with zero production cost (e.g. selling a media file on iTunes). Every time a user visits the site, you offer a price p, and the user makes a purchase (and pays p) if and only if p is less than or equal to the maximum amount the user is willing to pay. This is a multi-armed bandit problem in which $\mathscr{S}$ is the set of possible prices (presumably $\mathbb{R}_+$) and the payoff of charging price p at time t is 0 if the user is unwilling to pay that price, or p if the user is willing to pay the price. Note that this is an example of a multi-armed bandit problem in which it is more natural to model the strategy set as a one-dimensional continuum than as a finite set.

Overlay routing in networks. In an *overlay network*, two hosts s and t may communicate using a multi-hop path $s = v_0, v_1, \ldots, v_L = t$. Suppose that the network experiences point-to-point delays $d(v, w)$ which change unpredictably over time, and that s and t can only measure the end-to-end delay $\sum_{i=1}^{L} d(v_{i-1}, v_i)$ on a multi-hop path. Their goal in each time period is to choose a routing path whose delay is as small as possible. This situation can be modeled as a multi-armed bandit problem in which $\mathscr{S}$ is the set of L-hop paths from s to t, and the cost of a path $P \in \mathscr{S}$ is equal to its end-to-end delay. This is an interesting special case of the multi-armed bandit problem because the size of $\mathscr{S}$ is generally exponential in the size of the problem description. Nevertheless, we will see in a future lecture that it is possible to design an efficient algorithm for this special case of the bandit problem by exploiting correlations between the costs of different elements of $\mathscr{S}$ (i.e. overlay paths).

3 The "phased simulation" algorithm

In this section we describe a multi-armed bandit algorithm which we call "phased simulation" (or PSim, for short) because it is based on the idea of dividing the timeline $1, \ldots, T$ into consecutive phases, and simulating the best-expert problem which each phase of the timeline representing a single time step in the simulation.

The PSim algorithm uses a subroutine which is an algorithm for the best-expert problem. We will denote this subroutine by BEX; one may use any of the best-expert algorithms we have seen so far in this course, e.g. Hedge or FPL. Note that both Hedge and FPL actually refer to one-parameter families of algorithms with a preconfigured parameter ε. We will accordingly assume that the subroutine BEX also has a preconfigured parameter $\varepsilon > 0$, and we'll use the notation BEX(ε) unless the ε is clear from context. We will assume that the regret of BEX(ε) sastisfies

$$\hat{R}(\mathsf{BEX}(\varepsilon), t) \leq \varepsilon + O\left(\frac{\log n}{\varepsilon t}\right).$$

To verify that Hedge(ε) and FPL(ε) satisfy this bound, see the lecture notes from prior weeks. (In the lecture notes for the FPL algorithm, we used the notation FPL_μ rather than FPL(ε).)

The algorithm PSim is a multi-armed bandit algorithm with two unspecified parameters ε, P. It works as follows.

- Assume $T = PL$ for an integer $L > n$.
- Divide time into P consecutive phases of length L. Let ϕ_j be the set of time steps in phase j.
- Let τ_j be a random one-to-one function from $[n]$ to ϕ_j. Let

$$\begin{aligned}
\textsc{Explore}_i &= \bigcup_{j=1}^{P} \{\tau_j(i)\} \\
\textsc{Explore} &= \bigcup_{i=1}^{n} \textsc{Explore}_i \\
\textsc{Exploit} &= [T] \setminus \textsc{Explore}.
\end{aligned}$$

- Let $\hat{c}_j$ denote the function $\hat{c}_j(i) = c_{\tau_j(i)}(i)$.
- Let p_j denote the probability distribution on $[n]$ which is the output of BEX(ε) at time j, on input $\hat{c}_1, \ldots, \hat{c}_{t-1}$. Let $\hat{x}_j$ denote a random sample from p_j.
- Pick

$$x_t = \begin{cases} i & \text{if } t \in \textsc{Explore}_i \\ \hat{x}_j & \text{if } t \in \phi_j \text{ and } t \notin \textsc{Explore}. \end{cases}$$

4 Analysis of PSim

The idea of PSim is that each phase simulates one round of BEX, using a "simulated cost function" $\hat{c}_j$ which is an estimator of the average of the actual cost functions during phase j. If P is large enough, the average of all the simulated cost functions, $\frac{1}{P}\sum_j \hat{c}_j$, will be a very good approximation to the average of all the actual cost functions, $\frac{1}{T}\sum_t c_t$, and since BEX picks a sequence of strategies which is close to the best strategy and we are almost always following the advice of BEX, we are coming close to the performance of the best strategy.

To make this precise, we start by defining $\mathscr{F}_{<j}$ to be the set of all random variables observed by the algorithm before phase j. In other words, letting $\phi_{<j}$ denote the union of ϕ_s for $1 \leq s < j$, we define $\mathscr{F}_{<j}$ to consist of the following random variables: x_t and $c_t(x_t)$ for all $t \in \phi_{<j}$, and all random bits used by the algorithm prior to the start of phase j.

Lemma 1. *$\hat{c}_j$ is an unbiased estimator of the average cost $\bar{c}_j = \frac{1}{L}\sum_{t\in\phi_j} c_t$. More precisely,*

$$\mathbf{E}\left[\hat{c}_j \mid \mathscr{F}_{<j}\right] = \bar{c}_j.$$

Proof. The left and right sides are functions from $[n]$ to $[0,1]$. To prove they are equal, it suffices to evaluate both sides at an arbitrary $i \in [n]$ and show that the results are equal. Both sides are equal to the average value of $c_t(i)$ as t runs over ϕ_j; in one case this is by definition of $\bar{c}$, in the other case it is because $\tau_j(i)$ is uniformly distributed in ϕ_j and is independent of $\mathscr{F}_{<j}$. □

Lemma 2. *For any $x \in \mathscr{S}$,*

$$\mathbf{E}\left[\hat{c}_j(\hat{x}_j) \mid \mathscr{F}_{<j}\right] = \mathbf{E}\left[\bar{c}_j(\hat{x}_j) \mid \mathscr{F}_{<j}\right].$$

Proof.

$$\begin{aligned}
\mathbf{E}\left[\hat{c}_j(\hat{x}_j) \mid \mathscr{F}_{<j}\right] &= \mathbf{E}\left[\sum_x p_j(x)\hat{c}_j(x) \,\middle|\, \mathscr{F}_{<j}\right] \\
&= \sum_x p_j(x)\mathbf{E}\left[\hat{c}_j(x) \mid \mathscr{F}_{<j}\right] \\
&= \sum_x p_j(x)\bar{c}_j(x) \\
&= \mathbf{E}\left[\bar{c}_j(\hat{x}_j) \mid \mathscr{F}_{<j}\right].
\end{aligned}$$

□

Lemma 3. *For all $x \in \mathscr{S}$,*

$$\frac{1}{T}\mathbf{E}\left[\sum_{t\in \text{EXPLOIT}} (c_t(x_t) - c_t(x))\right] \leq \varepsilon + \frac{\log n}{\varepsilon P}.$$

Proof.

$$\begin{aligned}\frac{1}{T}\sum_{t\in\textsc{Exploit}}\mathbf{E}\left[c_t(x_t)-c_t(x)\right] &\leq \frac{1}{T}\sum_{t=1}^{T}\mathbf{E}\left[c_t(\hat{x}_j)-c_t(x)\right]\\ &= \frac{1}{P}\sum_{j=1}^{P}\mathbf{E}\left[\bar{c}_j(\hat{x}_j)-\bar{c}_j(x)\right]\\ &\leq \varepsilon+\frac{\log n}{\varepsilon P}.\end{aligned}$$

□

Lemma 4. *For all $x \in \mathscr{S}$,*

$$\frac{1}{T}\sum_{t\in\textsc{Explore}}\mathbf{E}\left[c_t(x_t)-c_t(x)\right]\leq\frac{nP}{T}.$$

Proof. For each $t \in$ Explore, $c_t(x_t) - c_t(x) \leq 1$. The lemma follows because Explore has nP elements. □

Theorem 5. *If $\varepsilon = (n\log(n)/T)^{1/3}$ and $P = \log^{1/3}(n)(T/n)^{2/3}$ then*

$$\hat{R}(\mathsf{PSim},T)=O\left(\left(\frac{n\log(n)}{T}\right)^{1/3}\right).$$

Proof. The preceding two lemmas make it clear that $\hat{R}(\mathsf{PSim},T)\leq\varepsilon+\frac{\log n}{\varepsilon P}+nP$. The rest of the proof consists of plugging in the given values of ε, P and checking that each term on the right side is equal to $(n\log(n)/T)^{1/3}$. □

4.1 The Exp3 algorithm

The preceding analysis of PSim is very simple and modular, which makes it easy to experiment with variations of the algorithm. For example, can we use phases of length $L = 1$? At first this seems ridiculous, because we need at least n steps in a phase so that we can explore each strategy at least once. But actually, we can build an unbiased estimator from a single measurement! First define a sampling rule for x_t as

$$x_t=\begin{cases}\text{sample uniformly} & \text{with probability } \gamma\\ \text{sample using } \mathsf{BEX} & \text{with probability } 1-\gamma\end{cases},$$

and let $\hat{p}_t$ denote the distribution induced by this sampling rule. Hence

$$\hat{p}_t(x)=(1-\gamma)p_t(x)+\gamma/n,$$

where p_t is the distribution coming from BEX. Let EXPLORE denote the set of steps in which the uniform sampling rule is used, and let EXPLOIT denote the set of steps in which the advice of BEX is used. Now define:

$$\hat{c}_t(x) = \begin{cases} c_t(x)/\hat{p}_t(x) & \text{if } x = x_t \\ 0 & \text{otherwise} \end{cases} .$$

The following lemmas are now easy. The proofs of three of them are omitted because they are so similar to the proofs in the preceding section.

Lemma 6. $\mathbf{E}(\hat{c}_t \mid \mathscr{F}_{<t}) = c_t$.

Lemma 7. $\mathbf{E}(\hat{c}_t(x_t) \mid \mathscr{F}_{<t}) = \mathbf{E}(c_t(x_t) \mid \mathscr{F}_{<t})$.

Lemma 8.

$$\frac{1}{T}\mathbf{E}\left[\sum_{t \in \text{EXPLOIT}} (c_t(x_t) - c_t(x))\right] < \frac{\varepsilon n}{\gamma} + \frac{n \log n}{\varepsilon \gamma T}.$$

Proof. The assumption that BEX has regret bounded by $\varepsilon + \log(n)/(\varepsilon T)$ applies only when costs are between 0 and 1. Here, the simulated costs $\hat{c}_t(x)$ are bounded above by n/γ, so we have to scale everything up by that amount. □

Lemma 9. $\frac{1}{T}\mathbf{E}\left[\sum_{t \in \text{EXPLORE}} (c_t(x_t) - c_t(x))\right] < \gamma$.

Proof. The expected number of t in EXPLORE is γT. □

Theorem 10. *If* $\gamma = (n \log(n)/T)^{1/4}$ *and* $\varepsilon = \gamma^2$ *then* $\hat{R}(\mathsf{Exp3}, T) = O(\gamma)$.

Proof. Sum up the bounds from the preceding two lemmas, and plug in the specified values of γ and ε. □

4.2 Improved analysis of Exp3

It turns out that if BEX is implemented as Hedge(γ/n), a much better bound can be proven. The secret is to improve Lemma 8.

Lemma 11. *If* $0 < \alpha < 1$ *and* $\hat{c}_t : [n] \to \mathbb{R}_+$ *then*

$$\sum_{t=1}^{T} \sum_{x \in [n]} p_t(x)\hat{c}_t(x) \le \max_{x \in [n]} \sum_{t=1}^{T} \hat{c}_t - \log(1-\alpha) \sum_{t=1}^{T} \sum_{x \in [n]} p_t(x)(\hat{c}_t(x))^2 + \frac{\log(n)}{\alpha}.$$

Proof. The proof is in the same format as the original analysis of Hedge: the log of the sum of the weights can be bounded from below by the weight of the best strategy, and it can be bounded from above by a function related to the algorithm's expected performance. Here there's a new twist: we need an upper bound on $(1-\alpha)^x$, but we can't use $1-\alpha x$ as before because this upper bound is valid only when $0 \le x \le 1$. Instead we use the degree-2 Taylor expansion of the function $(1-\alpha)^x$. We claim that for all $x \ge 0$,

$$(1-\alpha)^x \le 1 + \log(1-\alpha)x + \log^2(1-\alpha)x^2.$$

To prove it, let $f(x)$ denote the left side and let $g(x)$ denote the right side. We have

$$\begin{aligned} f(0) = g(0) &= 0 \\ f'(0) = g'(0) &= \log(1-\alpha) \\ f''(0) = \log^2(1-\alpha) &< 2\log^2(1-\alpha) = g''(0). \end{aligned}$$

In fact for all $x > 0$ we have

$$f''(x) < f''(0) < g''(0) = g''(x).$$

So the function $h(x) = g(x) - f(x)$ satisfies $h(0) = h'(0) = 0$ and $h''(x) > 0$ for all $x \ge 0$. This implies $h(x) > 0$ for all $x > 0$ which establishes the claim.

Putting this claim to use, let's define

$$w_t(x) = (1-\alpha)^{\sum_{s=1}^{t} \hat{c}_s(x)}$$

and

$$W_t = \sum_x w_t(x).$$

We have $p_t(x) = w_{t-1}(x)/W_{t-1}$, and

$$\begin{aligned} \frac{W_t}{W_{t-1}} &= \sum_x p_t(x)(1-\alpha)^{\hat{c}_t(x)} \\ &\le 1 + \log(1-\alpha)\sum_x p_t(x)\hat{c}_t(x) + \log^2(1-\alpha)\sum_x p_t(x)(\hat{c}_t(x))^2 \\ \log(W_t) - \log(W_{t-1}) &\le \log(1-\alpha)\sum_x p_t(x)\hat{c}_t(x) + \log^2(1-\alpha)\sum_x p_t(x)(\hat{c}_t(x))^2 \\ \log(W_T) - \log(n) &\le \log(1-\alpha)\sum_{t,x} p_t(x)\hat{c}_t(x) + \log^2(1-\alpha)\sum_{t,x} p_t(x)(\hat{c}_t(x))^2 \end{aligned}$$

We also have $\log(W_T) \ge -\log(1-\alpha)\sum_{t=1}^{T} \hat{c}_t(x)$. The lemma follows by rearranging terms and using the identity $-1/(\log(1-\alpha)) \le 1/\alpha$. □

Lemma 12. *If* $0 < \gamma \le 1/2$ *and* Hedge(γ/n) *is used as the* BEX *subroutine in* Exp3*, then*

$$\frac{1}{T}\mathbf{E}\left[\sum_{t \in \text{EXPLOIT}} c_t(x_t) - c_t(x)\right] \le (4\ln 2)\gamma T + \frac{n\log n}{\gamma}$$

Proof. Lemma 11 implies that the left side is bounded above by

$$-\log(1-\gamma/n)\mathbf{E}\left[\sum_{t,x} p_t(x)(\hat{c}_t(x))^2\right] + \frac{n\log(n)}{\gamma}.$$

For $0 < \gamma/n \leq 1/2$ we have $-\log(1-\gamma/n) \leq (2\ln 2)\gamma/n$. We also have $p_t(x) = \frac{1}{1-\gamma}\left(\hat{p}_t(x) - \gamma/n\right) < 2\hat{p}_t(x)$, hence

$$p_t(x)\hat{c}_t(x) \leq 2\hat{p}_t(x)\hat{c}_t(x) \leq 2.$$

Thus

$$\begin{aligned}
-\log(1-\gamma/n)\mathbf{E}\left[\sum_{t,x} p_t(x)(\hat{c}_t(x))^2\right] &< (4\ln 2)\frac{\gamma}{n}\sum_{t,x}\mathbf{E}\left[\hat{c}_t(x)\right] \\
&= (4\ln 2)\frac{\gamma}{n}\sum_{t,x} c_t(x) \\
&\leq (4\ln 2)\frac{\gamma}{n}(nT) \\
&= (4\ln 2)\gamma T.
\end{aligned}$$

□

Theorem 13. *The regret of* Exp3 *is* $O(\sqrt{Tn\log n})$.

Proof. Put $\gamma = \sqrt{n\log n/T}$ and check that each term in Lemma 9 and Lemma 12 is $O(\sqrt{Tn\log n})$. □