CMSC 858G: Bandits, Experts and Games 11/03/16

# Lecture 9: (Semi-)bandits and experts with linear costs (part I)

Instructor: Alex Slivkins Scribed by: Amr Sharaf

In this lecture, we will study bandit problems with linear costs. In this setting, actions are represented by vectors in a low-dimensional real space. For simplicity, we will assume that all actions lie within a unit hypercube: $a \in [0,1]^d$. The action costs $c_t(a)$ are linear in the vector $a$, namely: $c_t(a) = a \cdot v_t$ for some weight vector $v_t \in \mathbb{R}^d$ which is the same for all actions, but depends on the current time step. This problem is useful and challenging under full feedback as well as under bandit feedback; further, we will consider an intermediate regime called *semi-bandit feedback*.

The plan for today is as follows:

1. motivate and define new bandit problems which fall under "linear bandits";
2. tackle this problem using the reduction from previous class (from bandit feedback to full feedback), augmented with some new tricks. Plugging `Hedge` into this reduction immediately gives a meaningful result for the new problems;
3. introduce a new algorithm for linear bandits with full feedback (*Follow the Perturbed Leader*), This algorithm is one of the fundamental results in the online learning literature, and (among many other applications) it plugs nicely into the above reduction, yielding a substantial improvement for the new problems.

## 1 Recap from last lecture

Let us recap some results from last lecture, restating stating them in a slightly more generic way.

As before, we have $K$ different actions and fixed time horizon $T$. Each action has an associated cost $c_t(a)$ at round $t$. The total cost of each action is the sum over all rounds:

$$\texttt{cost}(a) = \Sigma_{t=1}^{T} c_t(a).$$

Regret is defined as the difference between the total cost of the algorithm and the minimum total cost of an action:

$$R(T) = \texttt{cost}(\texttt{ALG}) - \min_a \texttt{cost}(a).$$

(We do not take expectations in this definition; instead, we upper-bound the expected regret.)

We have algorithm `Hedge` for the full-feedback setting. Against an adaptive randomized adversary, it achieves regret

$$\mathbb{E}[R(T)] \leq \theta(\sqrt{UT \log K}), \tag{1}$$

where $U$ is some known upper bound on the action costs, i.e. $U \geq c_t(a)$.

We will also use a reduction from bandit feedback to full feedback: *i.e.,* this reduction uses a best-expert algorithm (for concreteness, `Hedge`) as a subroutine, and constructs a bandit algorithm.

**Algorithm 1** Reduction from bandit feedback to full feedback.

**Given:** best-experts algorithm `ALG` and parameter $\gamma \in (0, \frac{1}{2})$.
In each round $t$:

1. call `ALG`, receive an expert $x_t$ chosen for this round (internally drawn from some distribution $p_t$ over the experts).
2. with probability $1 - \gamma$ follow expert $x_t$; else chose arm via "random exploration" (TBD)
3. observe cost $c_t$ for the chosen arm, and perhaps some extra information (TBD)
4. define "fake costs" $\hat{c}_t(x)$ for each expert $x$ (TBD), and return them to `ALG`.

The reduction creates an expert for each arm: this expert always recommends this arm. The bandit algorithm proceeds as shown in Algorithm 1.

While several steps in the algorithm are unspecified, the analysis from last lecture applies word-by-word even at this level of generality: *i.e., no matter how these missing steps are filled in.*

**Theorem 1.1.** *Assume deterministic, oblivious adversary. Assume "fake costs" are defined so that* $\mathbb{E}[\hat{c}_t(x)|p_t] = c_t(x)$ *and* $\hat{c}_t(x) \leq U'/\gamma$ *for all experts* $x$ *and all rounds* $t$*, where* $U'$ *is some number that is known to the algorithm. Then the reduction (Algorithm 1) with best-experts algorithm* `Hedge` *and an appropriate choice of parameter* $\gamma$ *(in terms of* $K, T, U'$*) achieves regret*

$$\mathbb{E}[R(T)] \leq O(T^{2/3})(U' \log K)^{1/3}.$$

*More generally,* $\mathbb{E}[R(T)]$ *is at most the expected regret of* `ALG` *(on fake costs) plus* $\gamma T$*.*

Later in this lecture, we will instantiate this algorithm (*i.e.,* specify the missing pieces) to obtain a solution for combinatorial semi-bandits.

## 2 The Online Routing Problem

In the *online routing problem* (a.k.a. *online shortest paths* problem), we are given a graph $G$ with $d$ edges, a source node $s$, and a target destination node $t$. The graph can either be directed or undirected. We have costs on edges that we interpret as delays in routing, or lengths in a shortest-path problem. The cost of a path is the sum over all edges in this path. The costs can change over time. Informally, the algorithm's goal in each round is to find the "best route" from $s$ to $t$: an $s$-$t$ path with minimal cost. Accordingly, we interpret it as a route that minimizes the total travel time from $s$ to $t$.

Thus, an algorithm chooses among "actions" that correspond to $s$-$t$ paths in the graph. To cast this problem as a special case of "linear bandits", observe that each path is can be specified by a subset of edges, which in turn can be specified by a $d$-dimensional binary vector $a \in \{0,1\}^d$. Here edges of the graph are numbered from 1 to $d$, and for each edge $e$ the corresponding entry $a_e$ equals 1 if and only if this edge is included in the path. Let $v_t = (c_t(e) : \text{edges } e \in G)$ be the vector of edge costs at round $t$. Then the cost of a path can be represented as a linear product $c_t(a) = a \cdot v_t = \sum_{e \in [d]} a_e \, c_t(e)$.

More formally, each round $t$ in this problem proceeds as follows:

1. Adversary chooses costs $c_t(e) \in [0,1]$ for all edges $e$.
2. Algorithm chooses $s$-$t$-path $a_t \subset \texttt{Edges}(G)$.
3. Incurs cost $c_t(a_t) = \Sigma_{e \in a_t} a_e \cdot c_t(e)$ and receives feedback.

There are three version of the problem, depending on which feedback is received:

- bandit feedback: only $c_t(a_t)$ is observed;
- semi-bandit feedback: costs on all edges in $a_t$ are observed;
- full-feedback: cost on all edges are observed

We will mainly focus on the semi-bandit version, assuming the costs are selected by a deterministic oblivious adversary.

As a preliminary attempt, observe that we can use `Exp3` algorithm for this problem, but both the regret as well as the runtime performance will be bad, since the regret in `Exp3` is proportional to square root of the number of actions, which in this case may be exponential in $d$. Instead, we seek a regret bound of the form:

$$\mathbb{E}[R(T)] \leq \texttt{poly}(d) \cdot T^{\beta}, \quad \text{where } \beta < 1.$$

To this end, we will use the reduction (Algorithm 1) with best-expert algorithm `Hedge`. The "extra information" in this algorithm is the semi-bandit feedback. We will need to specify two other missing steps: the "random exploration" and the "fake costs".

Without loss of generality, assume that each edge $e$ belongs to some $s$-$t$ path $a^{(e)}$ (else, we can just remove this edge from the graph). For the "random exploration step", instead of selecting an action uniformly at random (as we did in `Exp4`), we select an edge $e$ uniformly at random, and pick the corresponding path $a^{(e)}$ as the chosen action.

To define fake costs, let $\Lambda_{t,e}$ be the event that in round $t$, the algorithm chooses "random exploration", *and* in random exploration, it chooses edge $e$. Note that $\Pr[\Lambda_{t,e}] = \gamma/d$. We define fake costs for each edge $e$ separately:

$$\hat{c}_t(e) = \begin{cases} \frac{c_t(e)}{\gamma/d} & \text{if event } \Lambda_{t,e} \text{ happens} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The fake cost of a path is simply the sum of fake costs over its edges. This completes the specification of an algorithm for the online routing problem with semi-bandit feedback; we will refer to this algorithm as `AlgSB`.

As in the previous lecture, we prove that fake costs provide unbiased estimates for true costs:

$$\mathbb{E}[\hat{c}_t(e) \mid p_t] = c_t(e) \quad \text{for each round } t \text{ and each edge } e.$$

Since the fake cost for each edge is at most $d/\gamma$, it follows that $c_t(a) \leq d^2/\gamma$ for each action $a$. Thus, we can immediately use Theorem 1.1 with `Hedge` and $U' = d^2$. For the number of actions, let us use an upper bound $K \leq 2^d$. Then $U' \log K \leq d^3$, and so:

**Theorem 2.1.** *Consider the online routing problem with semi-bandit feedback. Assume deterministic oblivious adversary. Algorithm* `AlgSB` *achieved regret* $\mathbb{E}[R(T)] \leq O(d\, T^{2/3})$.

*Remark* 2.2. Fake cost $\hat{c}_t(e)$ is determined by the corresponding true cost $c_t(e)$ and event $\Lambda_{t,e}$ which does not depend on algorithm's actions. Therefore, fake costs are chosen by a (randomized) oblivious adversary. In particular, in order to apply Theorem 1.1 with a different best-experts algorithm `ALG`, it suffices to have an upper bound on regret against an oblivious adversary.

# 3 Combinatorial (semi-)bandits

The online routing problem discussed above is a special case of *combinatorial semi-bandits*, where edges are replaced with $d$ "atoms", and $s$-$t$ paths are replaced with feasible subsets of atoms. The family of feasible subsets can be arbitrary.[1]

The algorithm and analysis from the previous section does not rely on any special properties of $s$-$t$ paths, and carry over word-by-word to combinatorial semi-bandits (replacing edges with atoms, and $s$-$t$ paths with feasible subsets). Therefore:

**Theorem 3.1.** *Consider combinatorial semi-bandits with deterministic oblivious adversary. Algorithm* `AlgSB` *achieved regret* $\mathbb{E}[R(T)] \leq O(d\,T^{2/3})$.

Let us list a few other notable special cases of combinatorial semi-bandits:

- *News Articles:* a news site needs to select a subset of articles to display to each user. The user can either click on an article or ignore it. Here, rounds correspond to users, atoms are the news articles, the reward is 1 if it is clicked and 0 otherwise, and feasible subsets here can encode a complicated set of constraints on selecting the articles.

- *Ads:* a website needs select a subset of ads to display to each user. For each displayed ad, we observe whether the user clicked on it, in which case the website receives some payment. The payment may, in principle, depend on an ad and on the user. Again: rounds correspond to users, atoms are the ads, and feasible subset can encode constraints on which ads can or cannot be shown together.

- *Network broadcast:* In each round, we want to transmit a packet from a source to multiple targets in the network. Then the feasible subsets correspond to (feasible) spanning subtrees for sending the packets.

- *A slate of news articles:* Similar to the news articles problem, except that ordering of the articles on the webpage matters. So the news site needs to select a slate (an ordered list) of articles. To represent this problem as an instance of combinatorial semi-bandits, define "atoms" to mean "this news article is chosen for that slot"; a subset of atoms is feasible if it defines a valid slate (*i.e.,* there is exactly one news article assigned to each slot).

Thus, combinatorial semi-bandits is a general setting which captures many motivating examples, and allows for a unified solution. Even better, this solution is itself an application of a more general framework. Such results are valuable even if each of the motivating examples is only a very idealized version of reality (*i.e.,* it captures some features of reality but ignores some others).

[1] Similarly, *combinatorial bandits* is the same problem, but with bandit feedback.

*Remark* 3.2. Solving the same problem with bandit feedback, known as *combinatorial bandits*, requires more work. The main challenge is that we need to estimate fake costs for all edges (resp., atoms) in the chosen path, whereas we only observe the total cost for the path. One solution is to construct a *basis*: a subset of feasible paths (called *base paths*) such that each edge can be represented as a linear combination of base paths. Then we can use a version of Algorithm 1 where in the "random exploration" step we chose uniformly among paths in this basis. This gives us fake costs on the base paths. Then fake cost on each edge can be defined as the corresponding linear combination over the base paths. This approach works as long as the linear coefficients are small. For more details, refer to Awerbuch and Kleinberg (2008).

Thus, we have an algorithm for combinatorial semi-bandits that achieves a good regret boun. However, the running time is slow because it relies on `Hedge` with a very large number of experts. We would like to design a faster algorithm, so that the runtime per time step polynomial in $d$ (rather than exponential in $d$).

Even if the costs on all atoms were known, to choose the best feasible action the algorithm would need to solve a combinatorial optimization problem: find a feasible subset with minimal cost. This problem is NP-hard in general, but allows polynomial-time solutions in many interesting special cases of combinatorial semi-bandits. For example, in the online routing problem discussed above the corresponding combinatorial optimization problem is a well-known shortest-path problem.

Thus, we should not hope to have a fast algorithm for the full generality of combinatorial semi-bandits. Instead, we assume that we have access to an *oracle*: an algorithm which finds the best feasible action given the costs on all atoms, and express the running time of our algorithm in terms of the number of oracle calls.

We will use this oracle to construct a new algorithm for combinatorial bandits with full feedback. In each round, this algorithm inputs only the costs on the atoms, and makes only one oracle call. It achieves regret $\mathbb{E}[R(T)] \leq O(U\sqrt{dT})$ against an oblivious adversary, where $U$ is a known upper bound on the costs of each expert in each round. The algorithm is called *Follow the Perturbed Leader* (`FPL`).

To solve the combinatorial semi-bandits, we use algorithm `AlgSB` as before, but replace `Hedge` with `FPL`; call the new algorithm `AlgSBwithFPL`. The same analysis from Section 2, applied to `AlgSBwithFPL`, yields regret

$$\mathbb{E}[R(T)] \leq O(U\sqrt{dT}) + \gamma T,$$

where $U = d^2/\gamma$ is an upper bound on the fake costs. Optimizing the choice of parameter $\gamma$, we immediately obtain the following theorem:

**Theorem 3.3.** *Consider combinatorial semi-bandits with deterministic oblivious adversary. Then algorithm* `AlgSBwithFPL` *with appropriately chosen parameter $\gamma$ achieved regret*

$$\mathbb{E}[R(T)] \leq O\left(d^{5/4}\, T^{3/4}\right).$$

*Remark* 3.4. In terms of the running time, it is essential that the fake costs on atoms can be computed *fast*: this is because the normalizing probability in (2) is known in advance.

Alternatively, we could have defined fake costs as

$$\hat{c}_t(e) = \begin{cases} \frac{c_t(e)}{q_t(e|p_t)} & \text{if } e \in a_t \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $q_t(e \mid p_t) = \Pr[e \in a_t \mid p_t]$. This definition allows for the same regret bound (and, in fact, is somewhat better in terms of regret), but requires computing $q_t(e \mid p_t)$. (At least naively) this computation requires looking at probabilities for all actions, which leads to running times exponential in $d$, similar to `Hedge`.

## References

Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *J. of Computer and System Sciences*, 74(1):97–114, February 2008. Preliminary version in *36th ACM STOC*, 2004.