CIS 700 Differential Privacy in Game Theory and Mechanism Design February 28, 2014

Lecture 7

Lecturer: Jamie Morgenstern *Scribe: Jamie Morgenstern*

Guest Lecture: Privacy-Preserving Public Information for Sequential Games

1 Introduction

Suppose a collection of strategic agents are trying to make some decision. For example, a collection of investment banks are trying to decide how to invest their money. The payoffs of the players will depend upon the actions of the other players; as more people choose a particular investment, the value to investers who are "later to the game" will be smaller than the value for earlier investors. If players make these decisions in some order, and have perfect information about earlier players' decisions, one can analyze the social welfare of various strategies or equilibrium concepts. In particular, if the utility of a player does not depend on later players, only upon the decisions made by earlier players, the greedy strategy is a dominant strategy (and the only undominated strategy). On the other hand, banks may not want their investment decisions to be public knowledge for their competition, which motivates looking into what happens if players are only provided approximate, *private* information regarding previous players' actions. In our work, we ask how well different strategies perform with respect to this approximate information. In particular, today I'll talk about how one can show that players playing greedily respect to approximate information gives social welfare whose ratio with the social wefare of OPT is bounded.

I'll start by describe a classic result from online optimization, and show how it extends to the setting where the information provided to distributed optimizers is only approximate. I'll then show an improved privacy-preserving counter scheme which gives better approximation guarantees in our settings, by combining additive and multiplicative approximation guarantees.

2 Max-weight matching: Greedy is a 2-approximation

Suppose n nodes on side U of a bipartite graph arrive online. These nodes arrive with unweighted edges to some subset of the m weighted vertices on the V side of the bipartite graph. Let the weight of a perfect matching be the sum of the weighted vertices in V who are matched to a vertex in U.

There is some optimal matching in the offline optimization setting: when all nodes and their respective edges are visible, there is a well-defined largest weight any perfect matching can achieve.

There is a well-known argument showing that the online greedy matching algorithm 2-approximates the maximum weight matching.

Theorem 1 ([5]) *The online matching algorithm which matches vertices in U greedily with weighted vertices in V is a 2-approximation to the optimal matching.*

Proof Consider any vertex $u_i \in U$ which greedy matches to some $v \in V$ where $v \neq v_i$, where v_i is the vertex u_i is matched to in OPT. u_i is matched to v for one of two reasons: either v_i was already matched previously, or $w(v) > w(v_i)$.

If v_i was matched previously, to some vertex u_j, charge $w(v_i)$ to the edge (u_j, v_i) (to the edge which caused u_i to pick a different match). If $w(v) > w(v_i)$, then charge $w(v_i)$ to the edge (u_i, v): charge the edge "responsible" for the deviation away from the edge (u_i, v_i) in OPT.

How much weight can be charged to a given edge in the greedy matching? An edge (u_i, v_j) can be charged for one of two reasons, either because u_j didn't have v_j available, and they were charged $w(v_j)$, or by u_i with weight $w(v_i) < w(v_j)$. Either case can happen at most once for a given edge (the edge can only take the endpoint v_j away from one vertex, and only replaces the edge (u_i, v_i) for i), and either case charges at most $w(v_j)$. Thus, each edge (u_i, v_j) in the greedy matching is charged at most twice $w(v_j)$, and so $Greedy \geq \frac{1}{2}OPT$. ■

This proof can easily be extended to the approximately greedy algorithm: suppose vertices u_i which arrive online choose to match with a v_j such that $w(v_j) \geq \frac{1}{\alpha}w(g_j)$, where g_j is the greedy choice for player j. Call the set of strategies which satisfy this constraint α-approximately greedy. Then, we have the following theorem.

Remark This result actually extends to players being allowed to pick *subsets* of their edges, so long as the set's they are allowed to pick are downward closed (e.g., if X is an allowed set of neighbors for a node, then $X' \subseteq X$ is also allowed), losing another factor of 2 in the approximation. Roughly speaking, either half of the utility from a set X was already taken by some other nodes, or half the utility still remains for X. The union of these facts implies a 4-approximation.

Remark This actually holds for the continuous version of this problem, where nodes can take nondiscrete pieces of nodes on the other side of the graph.

3 Approximate information: enough for online vertex-weighted matching?

Notice that, with no information, "greedy behavior" with respect to initial values can do quite poorly (there can be $\Omega(n)$ ratio between OPT and this behavior). We ask what one can do with private information.

We identify the nodes on the right-hand side of the bipartite graph with the investment opportunities, or resources, from the introduction. Suppose that a given resource r has k_r "copies", that is, the first k_r people to select r get utility v_r, and no other players get utility from choosing r. In this section, we prove that approximate information about how many people have selected a given resource or investment is enough for players to approximate OPT with greedy behavior, for arbitrary v_r, k_r. In fact, all of the results in this section actually apply in much more general settings, where each resource r has diminishing value for each subsequent "copy" of the resource chosen by a player (see the full paper for more details).

It is interesting to note that, while the counts are approximate, the value each individual player is getting isn't well-approximated. That is, an individual might have an unbounded ratio between her perceived utility from picking resource r (if no more copies exist but the counters say there are still copies available). The welfare approximation is only true when summed over all players' utilities.

Before presenting the theorem for this section, we need a definition of *approximate counters* in the continual observation setting. We will say a mechanism $\mathbf{s}$ provides an (α, β, γ)-approximate counter vector with respect to an input stream $a = (\mathbf{a}^1, \ldots, \mathbf{a}^n)$ where $\mathbf{a}^i = (a^i_1, \ldots, a^i_m)$ if, with probability $1 - \gamma$, for all i, for all r, it is the case that

$$\frac{1}{\alpha}x^i_r - \beta \leq y^i_r \leq \alpha x^i_r + \beta$$

where $x^i_r \sum_{j=1}^{i-1} a^i_r$ is the partial sum for resource r. That is, the guarantee is that all the approximate partial sums are correct up to an additive factor β and a multiplicative factor α within the true partial sum values.

The main theorem we will prove for the simpler case of k_r identical copies of r each with value v_r, is the following.

Theorem 2 *Suppose y is (α, β, γ)-approximate counter vector which only underestimates x^i_r for all i, r. Then, the social welfare resulting from players play greedily with respect to y in the online vertex-weighted matching setting is at least an X-fraction of the social welfare of OPT, with probability $1 - \gamma$.*

We make the following claim before proving Theorem 2.

Claim 1 *Suppose k players select r according to greedy behaviour with respect to an $(\alpha, \beta, 0)$-approximate counter vector and expect to get nonzero utility. Then,*

$$\frac{k}{\min(k_r, k)} = \frac{\textit{Perceived utility from } r}{\textit{Actual welfare from } r} \leq \alpha(1+\beta)$$

Now, with Claim 1 in hand, we prove Theorem 2. **Proof** [Proof of Theorem 2]
The counters are accurate with probability $1-\gamma$, so with that probability Claim 1 holds. Assume we are in that case.

Let $SW(Greedy_{counters})$ denote the social welfare of players playing greedily with respect to the counter values. Let $PSW(Greedy_{counters})$ denote the *perceived* social welfare of players playing greedily with respec to the counters (e.g, if $s_r^i < k_r$, then player i choosing resource r has perceived utility v_r). Let $PSW(OPT_{counters})$ denote the optimal allocation if the counters' values were accurate, (e.g. there are actually k_r + number of people who see $s_r^i < k_r$ when $x_r^i \geq k_r$ copies of resource r). Finally, let $SW(OPT_{Real})$ denote the social welfare of the optimal solution with k_r copies of resource r. Then, we have

$$SW(Greedy_{counters}) \geq \frac{PSW(Greedy_{counters})}{\alpha(1+\beta)} \geq \frac{PSW(OPT_{counters})}{2\alpha(1+\beta)} \geq \frac{SW(OPT_{Real})}{2\alpha(1+\beta)}$$

The first inequality comes from Claim 1, the second from the fact that greedy is a 2-approx to OPT on any graph, and the third comes from from the fact that the counters are undercounting (e.g., that there are weakly more copies of resource r in the perceived graph than in the true graph), so $OPT_{counters}$ has more to choose between and thus a weakly larger optimum.

■

We still need to prove the claim we used in the previous proof. **Proof** [Proof of Claim 1]
Consider the k people who chose resource r. If $k_r \geq k$, then

$$\frac{k}{\min(k_r, k)} = \frac{k}{k} = 1$$

Thus, the only interesting case is where $k_r < k$. Now, if $k_r < k$, we have

$$\frac{k}{\min(k_r, k)} = \frac{k}{k_r}$$

and we need to bound this ratio. In particular, how large can x_r^i be if $y_r^i < k_r$? We know that

$$k_r \geq y_r^i \geq \frac{1}{\alpha} x_r^i - \beta$$

or, equivalently, that $\alpha(k_r + \beta) \geq x_r^i$. Thus,

$$\frac{k}{k_r} \leq \frac{\alpha(k_r + \beta)}{k_r} \leq \alpha + \alpha\beta$$

as desired. ■

4 A better Privacy-preserving mechanism

Theorem 2 motivates looking for counter schemes that can get slightly better additive approximations at the expense of a small multiplicative loss in accuracy; the guarantees of the theorem degrade linearly with respect to $\alpha\beta$. So, if $\alpha = O(1)$, then $\beta = o(log^2(n))$ would give a better approximation guarantee than the best-known counter mechanism (Chan et al or Dwork et al's binary tree-sum protocol). Here, we present a mechanism with improved additive guarentees at the expense of an (arbitrarily small) multiplicative loss in accuracy.

Recall the basic counter problem: given a stream $\mathbf{a} = (a_1, a_2, ..., a_n)$ of numbers $a_i \in [0, 1]$, we wish to release at every time step t the partial sum $x_t = \sum_{i=1}^{t} a_i$.

We require a generalization, where one maintains a vector of m counters. Each player's update contribution is now a vector $a_i \in [0, 1]^m$, with the constraint that $\|a_i\|_1 \leq 1$. That is, a player can add non-negative values to all counters, but the total value of her updates is at most 1. The partial sums x_t then lie in $(\mathbb{R}^+)^m$ (with ℓ_1 norm bounded by t).

Given an algorithm $\mathcal{A}$, we define the output stream $(s_1, s_2, ..., s_n) = \mathcal{A}(\mathbf{a})$ where $s_i = \mathcal{A}(t, a_1, ..., a_{i-1})$. The original works on differentially private counters [3, 2] concentrated on minimizing the additive error of the estimated sums, that is, they sought to minimize $\|x_t - s_t\|_\infty$. Both papers gave a binary tree-based mechanism, which we dub "TreeSum", with additive error approximately $(\log^2 n)/\epsilon$. Some of our algorithms use TreeSum, and others use a new mechanism (FTSum, described below) which gets a better additive error guarantee at the price of introducing a small multiplicative error. We capture a mixed approximation guarantee as follows:

Definition 2 *The algorithm $\mathcal{A}$ provides an (α, β, γ)-approximation to partial sums if for every (adaptively defined) sequence $\mathbf{a} \in ([0, 1]^m)^n$, with probability at least $1 - \gamma$ over the coins of $\mathcal{A}$, for all times $i \in [n]$ and counters $r \in [m]$, the reported value $x_{t,r}$ satisfies:*

$$\frac{1}{\alpha} \cdot x_{i,r} - \beta \leq s_{i,r} \leq \alpha \cdot x_{i,r} + \beta\,.$$

Proofs of all the results in this section can be found in Appendix 5.

Lemma 3 *For every $m \in \mathbb{N}$ and $\gamma \in (0, 1)$: Running m independent copies of TreeSum [3, 2] is $(\epsilon, 0)$-differentially private and provides an $(1, C_{tree} \cdot \frac{(\log n)(\log(nm/\gamma))}{\epsilon}, \gamma)$-approximation to partial vector sums, where $C_{tree} > 0$ is an absolute constant.*

Even for $m = 1, \alpha = 1$, this bound is slightly tighter than those in [2] and [3]; however, it follows directly from the tail bound in [2].

Our new algorithm, FTSum (for Flag/Tree Sum), is described in Algorithm 1. For small m ($m = o(log(n))$), it provides lower additive error at the expense of introducing an arbitrarily small constant multiplicative error.

Lemma 4 *For every $m \in \mathbb{N}$, $\alpha > 1$ and $\gamma \in (0, 1)$, FTSum (Algorithm 1) is $(\epsilon, 0)$-differentially private and $(\alpha, \tilde{O}_\alpha(\frac{m \log(n/\gamma)}{\epsilon}), \gamma)$-approximates partial sums (where $\tilde{O}_a(\cdot)$ hides polylogarithmic factors in its argument, and treats α as constant).*

FTSum proceeds in two phases. In the first phase, it increments the reported output value only when the underlying counter value has increased significantly. Specifically, the mechanism outputs a public signal, which we will call a "flag", roughly when the true counter achieves the values $\log n$, $\alpha \log n$, $\alpha^2 \log n$ and so on, where α is the desired *multiplicative* approximation. The reported estimate is updated each time a flag is raised (it starts at 0, and then increases to $\log n$, $\alpha \log n$, etc). The privacy analysis for this phase is based on the "sparse vector" technique of [4], which shows that the cost to privacy is proportional to the number of times a flag is raised (but not the number of time steps between flags).

When the value of the counter becomes large (about $\frac{\alpha \log^2 n}{(\alpha-1)\epsilon}$), the algorithm switches to the second phase and simply uses the TreeSum protocol, whose additive error (about $\frac{\log^2 n}{\epsilon}$) is low enough to provide an α multiplicative guarantee (without need for the extra space given by the additive approximation).

If the mechanism were to raise a flag *exactly* when the true counter achieved the values $\log n$, $\alpha \log n$, $\alpha^2 \log n$, etc, then the mechanism would provide a $(\alpha, \log n, 0)$ approximation during the first phase, and a $(\alpha, 0, 0)$ approximation thereafter. The rigorous analysis is more complicated, since flags are raised only near those thresholds.

Algorithm 1: FTSum — A Private Counter with Low Multiplicative Error

```
Input: Stream $\mathbf{a} = (a_1, ..., a_n) \in ([0,1]^m)^n$, parameters $m, n \in \mathbb{N}$, $\alpha > 1$ and $\gamma > 0$
Output: Noisy partial sums $s_1, ..., s_n \in \mathbb{R}^m$
$k \leftarrow \lceil \log_\alpha(\frac{\alpha}{\alpha-1} \cdot C_{tree} \cdot \frac{\log(nm/\gamma)}{\epsilon}) \rceil$;
/* $C_{tree}$ is the constant from Lemma 3 */
$\epsilon' \leftarrow \frac{\epsilon}{2m(k+1)}$;
for $r = 1$ to $m$ do
    $\mathrm{flag}_r \leftarrow 0$;
    $x_{0,r} \leftarrow 0$;
    $\tau_r \leftarrow (\log n) + \mathrm{Lap}(2/\epsilon')$;
for $i = 1$ to $n$ do
    for $r = 1$ to $m$ do
        if $flag_r \leq k$ then (First phase still in progress for counter $r$)
            $x_{i,r} \leftarrow x_{i-1,r} + a_{i,r}$;
            $\tilde{x_{i,r}} \leftarrow x_{i,r} + \mathrm{Lap}(\frac{2}{\epsilon'})$;
            if $\tilde{x_{i,r}} > \tau_r$ then (Raise a new flag for counter $r$)
                $\mathrm{flag}_r \leftarrow \mathrm{flag}_r + 1$;
                $\tau_r \leftarrow (\log n) \cdot \alpha^{\mathrm{flag}_r} + \mathrm{Lap}(2/\epsilon')$;
            Release $s_{i,r} = (\log n) \cdot \alpha^{\mathrm{flag}_r - 1}$ ;
        else (Second phase has been reached for counter $r$)
            Release $s_{i,r} =$ $r$-th counter output from TreeSum($\mathbf{a}, \epsilon/2$));
```

Proposition 1 *If $\mathcal{A}$ is (ϵ, δ)-private and (α, β, γ)-accurate, then one can modify $\mathcal{A}$ to obtain an algorithm $\mathcal{A}'$ with the same efficiency that is $(\epsilon, \delta + \gamma)$-private and $(\alpha, \beta, 0)$-accurate.*

Corollary 5 *Algorithm 1 is an (ϵ, δ)-differentially private vector counter algorithm providing a*

1. $(1, O(\frac{(\log n)(\log(nm/\delta))}{\epsilon}), 0)$*-approximation (using modified TreeSum); or*
2. $(\alpha, \tilde{O}_\alpha(\frac{m \log n \log\log(1/\delta)}{\epsilon}), 0)$*-approximation for any constant $\alpha > 1$ (using FTSum).*

Bibliographic Information The results from this lecture are from [1]

References

[1] Avrim Blum, Jamie Morgenstern, Ankit Sharma, and Adam Smith. Privacy-preserving public information for sequential games. *arXiv preprint arXiv:1402.4488*, 2014.

[2] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26, 2011.

[3] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. STOC '10, pages 715–724. ACM, 2010.

[4] Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS '10*, 2010.

[5] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC '90*, pages 352–358, 1990.

[6] Aaron Roth. Cis 800/002, fall 2011: The algorithmic foundations of data privacy, 2011.

5 Analysis of Private Counters

Proof [of Lemma 3] We assume the reader is familiar with the TreeSum mechanism. The privacy of this construction follows the same argument as for the original constructions. One can view m independent copies of the TreeSum protocol as a single protocol where the Laplace mechanism is used to release the entire vector of partial sums. Because the ℓ_1-sensitivity of each partial sum is 1 (since $\|a_t\| \leq 1$), the amount of Laplace noise (per entry) needed to release the m-dimensional vector partial sums case is the same as for a dimensional 1-dimensional counter.

To see why the approximation claims holds, we can apply Lemma 2.8 from [2] (a tail bound for sums of independent Laplace random variables) with $b_1 = \cdots = b_{\log n} = \log n/\epsilon$, error probability $\delta = \gamma/mn$, $\nu = \frac{(\log n)\sqrt{\log(1/\delta)}}{\epsilon}$ and $\lambda = \frac{(\log n)(\log(1/\delta)}{\epsilon}$, we get that each individual counter estimate $s_t(j)$ has additive error $O(\frac{(\log n)(\log(nm/\gamma))}{\epsilon})$ with probability at least $1-\gamma/(mn)$. Thus, all $n{\cdot}m$ estimates satisfy the bound simultaneously with probability at least $1-\gamma$. ■

Proof [of Lemma 4] We begin with the proof of privacy. The first phase of the protocol is $\epsilon/2$-differentially private because it is an instance of the "sparse vector" technique of [4] (see also [6, Lecture 20] for a self-contained exposition). The second phase of the protocol is $\epsilon/2$-differentially private by the privacy of TreeSum. Since differential privacy composes, the scheme as a whole is ϵ-differentially private. Note that since we are proving $(\epsilon, 0)$-differential privacy, it suffices to consider nonadaptive streams; the adaptive privacy definition then follows [3].

We turn to proving the approximation guarantee. Note that the each of the Laplace noise variables added in phase 1 of the algorithm (to compute $\tilde{x_{t,r}}$ and τ_j) uses parameter $2/\epsilon'$. Taking a union bound over the mn possible times that such noise is added, we see that with probability at least $1-\gamma/2$, each of these random variables has absolute value at most $O(\frac{\log(mn/\gamma)}{\epsilon'}$. Since $\frac{2}{\epsilon'} = O(\frac{mk}{\epsilon})$ and $k = O(\log\log(\frac{nm}{\gamma}) + \log\frac{1}{\epsilon})$, we get that each of these noise variables has absolute value $\tilde{O}_\alpha(\frac{m\log(mn/\gamma)}{\epsilon})$ with probability all but $\gamma/2$. We denote this bound E_1.

Thus, for each counter, the i-th flag is raised no earlier than when the value of the counter first exceeds $\alpha^i(\log n) - E_1$, and no later than when the counter first exceeds $\alpha^i(\log n) + E_1$. The very first flag might be raised when counter has value 0. In that case, the additive error of the estimate is $\log n$, which is less than E_1. Hence, he mechanism's estimates during the first phase provide an $(\alpha, E_1, \gamma/2)$-approximation (as desired).

The flag that causes the algorithm to enter the second phase is supposed to be raised when the counter takes the value $A := \alpha^k(\log n) \geq \frac{\alpha}{\alpha-1} \cdot C_{tree} \cdot \frac{\log(nm/\gamma)}{\epsilon}$; in fact, the counter could be as small as $A - E_1$. After that point, the additive error is due to the TreeSum protocol and is at most $B := C_{tree} \cdot \log(n) \cdot \log(nm/\gamma)/\epsilon$ (with probability at least $1-\gamma/2$) by Lemma 3. The reported value $s_{i,r}$ thus satisfies

$$s_{i,r} \geq x_{i,r} - B = \frac{1}{\alpha}x_{i,r} + \underbrace{(1-\frac{1}{\alpha})x_{i,r} - B}_{\text{residual error}} .$$

Since $x_{i,r} \geq A - E_1$, the "residual error" in the equation above is at least $(1 - \frac{1}{\alpha})(A - E_1) - B = -(1 - \frac{1}{\alpha})E_1 \geq -E_1$. Thus, the second phase of the algorithm also provides $(\alpha, E_1, \gamma/2)$-approximation. With probability $1 - \gamma$, both phases jointly provide a (α, E_1, γ)-approximation, as desired. ■

.