CIS 800/002 The Algorithmic Foundations of Data Privacy October 27, 2011

Lecture 13

Lecturer: Aaron Roth *Scribe: Aaron Roth*

Efficient Interactive Query Release Mechanisms

Recall that last time we gave a reduction from database update algorithms to query release mechanisms in the interactive setting (We no longer need a distinguisher!). We proved the following theorem.

Theorem 1 *Together with a $B(\alpha)$-Database Update Algorithm, OnlineIC is $(3\alpha, \beta)$-useful for α such that:*

$$\alpha = \frac{4B(\alpha)(\log|\mathcal{C}| + \log(2/\beta))}{\epsilon n}$$

Moreover, this reduction is efficiency preserving! That is, the running time per update is at most the running time per update of the database update algorithm. All that remains to give a computationally efficient algorithm is to give a computationally efficient database update algorithm. This is precisely what we will do for the class of *sparse linear queries.*

Definition 2 (Sparsity) *The* sparsity *of a linear query $Q : X \to [0,1]$ is $|\{x \in X : Q(x) > 0\}|$, the number of elements in the universe on which it takes a non-zero value. We say that a query is m-sparse if its sparsity is at most m. We will also refer to the class of all m-sparse linear queries, denoted Q_m. We say that a query class is* sparse *if it is m-sparse for $m = poly(n)$.*

We will assume that given an m-sparse query, we can quickly (in time polynomial in m) enumerate the elements $x \in X$ on which $Q(x) > 0$.

We will give a variant on the Multiplicative Weights DUA for sparse queries that has running time which depends only on m for m-sparse queries, rather than on $|X|$. Surprisingly, the accuracy we obtain will also depend only on m and will be independent of $|X|$! That is, the mechanism will make sense even in an *infinite universe model*, in which there is no pre-specified bound on the size of the data universe.

The idea of the algorithm will be to run the multiplicative weights DUA on a smaller universe $\hat{X}$. Because the queries are sparse, we know that only a small number of universe elements will ever require an update. If we knew which these were, we could run the MW algorithm only on this subset! The problem is that we do not know which universe elements these may be ahead of time. The trick will be to defer the decision of which elements from X to map to $\hat{X}$ until the last minute, adaptively assigning elements to elements of $\hat{X}$ only as updates are performed. To do this, we run our algorithm on a datastructure which we call a sparse multiplicative weights datastructure.

Definition 3 (Sparse Multiplicative Weights Data Structure) *The sparse multiplicative weights data structure $\mathcal{D}$ of size s is composed of three parts. We write $\mathcal{D} = (D, h, ind)$.*

1. *D is a collection of s real valued variables $x_1, \ldots, x_s$, with $x_i \in [0,1]$ for all $i \in [s]$. Variable x_i for $i \in [s]$ is referenced by $D[i]$. Initially $x_i = 1/s$ for all $i \in [s]$. We define $D[i] = 0$ for all $i > s$.*

2. *h is a hash table $h : \mathcal{X} \to [s] \cup \emptyset$ mapping elements in the universe X to indices $i \in [s]$. Elements $x \in \mathcal{X}$ can also be unassigned in which case we write $h(x) = \emptyset$. Initially, $h(x) = \emptyset$ for all $x \in \mathcal{X}$*

3. *$ind \in [s+1]$ is a counter denoting the index of the first unassigned variable. For all $i < ind$, there exists some $x \in \mathcal{X}$ such that $h(x) = i$. For all $i \geq ind$, there does not exist any $x \in \mathcal{X}$ such that $h(x) = i$. Initially $ind = 1$.*

If $ind \leq s$, we can add *an unassigned element $x \in \mathcal{X}$ to $\mathcal{D}$. Adding an element $x \in \mathcal{X}$ to $\mathcal{D}$ sets $h(x) \leftarrow ind$ and increments $ind \leftarrow ind + 1$. If $ind = s + 1$, attempting to add an element causes the data structure to report **FAILURE**.*

A linear query Q is evaluated on a sparse MW data structure $\mathcal{D} = (D, h)$ as follows.

$$Q(\mathcal{D}) = \sum_{x \in \mathcal{X}: Q(x) > 0 \wedge h(x) \neq \emptyset} Q(x) \cdot D[h(x)] + \sum_{x \in \mathcal{X}: Q(x) > 0 \wedge h(x) = \emptyset} Q(x) \cdot D[ind]$$

We can now define the sparse multiplicative weights algorithm:

Algorithm 1 The Sparse Multiplicative Weights (SMW) IDC Algorithm for m-sparse queries. It is instantiated with an accuracy parameter $\eta = \alpha/2$. It takes as input a sparse MW datastructure $\mathcal{D}$, an m-sparse query $Q \in Q_m$, and an estimate of the query value v_t.

SMW$(\mathcal{D}_t = (D_t, h_t, \text{ind}_t), Q_t, v_t)$:

if $\mathcal{D}_t = \emptyset$ **then**

Let s be the smallest integer such that $s/(\log(s) + 1) \geq 4m/\alpha^2$.

Return a new Sparse MW data structure $\mathcal{D}_1 = (D_1, h_1, \text{ind}_1)$ of size s with $h_1(x) = \emptyset$ for all $x \in \mathcal{X}$, $x_i = 1/s$ for all $i \in [s]$, and $\text{ind}_1 = 1$.

end if

Let $\mathcal{D}_{t+1} = (D_{t+1}, h_{t+1}, \text{ind}_{t+1}) \leftarrow \mathcal{D}_t$

Update: For all $x \in \mathcal{X}$ such that $Q_t(x) > 0$: **If** $h_{t+1}(x) = \emptyset$ then **add** x to $\mathcal{D}_{t+1}$.

if $v_t < Q_t(\mathcal{D}_t)$ **then**

Update: For all $x \in \mathcal{X}$ such that $Q_t(x) > 0$: Let

$$D_{t+1}[h_{t+1}(x)] \leftarrow D_{t+1}[h_{t+1}(x)] \cdot \exp(-\eta Q_t(x))$$

else

Update: For all $x \in \mathcal{X}$ such that $Q_t(x) > 0$: Let

$$D_{t+1}[h_{t+1}(x)] \leftarrow D_{t+1}[h_{t+1}(x)] \cdot \exp(\eta Q_t(x))$$

end if

Normalize: For all $i \in [s]$:

$$D_{t+1}[i] = \frac{D_{t+1}[i]}{\sum_{j=1}^{s} D_{t+1}[j]}$$

Output $\mathcal{D}_{t+1}$.

Theorem 4 *The Sparse Multiplicative Weights algorithm is a $B(\alpha)$-DUA for the class of m-sparse queries Q_m, where:*

$$B(\alpha) = 4\frac{\log s}{\alpha^2}$$

and s is the smallest integer such that $s/(\log(s) + 1) \geq 4m/\alpha^2$.

Proof [Proof Sketch] The proof is a modification of the proof that the multiplicative weights mechanism is a $B(\alpha)$-DUA for $B(\alpha) = 4\log|X|/\alpha^2$, and is very similar. There are just a couple of hitches – in particular, we have to define a potential function with respect only to the elements in our smaller universe, which we have to be careful about because we are assigning elements to this universe adaptively. To handle this, we consider a different potential function for each maximal database update sequence, defined in terms of the assignments of universe elements in the last datastructure in the sequence. But there are no major surprises. We won't go over the details that part of the proof again, just the structure.

Briefly, We will consider any maximal (SMW, $\mathcal{D}, C, \alpha, L$)-database update sequence $\{(\mathcal{D}^t, Q_t, v_t)\}_{t=1,\ldots,L}$. We must argue that $L \leq \frac{4(\log s+1)}{\alpha^2}$ and that no data structure $\mathcal{D}^t$ in the sequence ever returns **FAILURE** when the SMW algorithm attempts to **add** some element $x \in X$ to it. Consider the real private

database D and the final data structure in the sequence $\mathcal{D}_T = (D_T, h_T, \text{ind}_T)$. We will define a non-negative potential function Ψ based on h_T and $\hat{D}$ and show that it decreases significantly at each step. We define:

$$\Psi_t \stackrel{\text{def}}{=} \sum_{x: h_T(x) \neq \emptyset} \hat{D}[x] \log \left(\frac{\hat{D}[x]}{D_t[h_T(x)]} \right)$$

Lemma 5 *For all $t \in [L]$, $\Psi_t \geq -\frac{1}{e}$ and $\Psi_0 \leq \log s$*

Proof Similar to before (with a couple of complicating details – i.e. observe that the potential function can now indeed be negative). ■

We will argue that in every step the potential drops by at least $\alpha^2/4$. Because the potential begins at $\log s$, and must always be non-negative, we therefore know that there can be at most $L \leq 4 \log s / \alpha^2$ steps. To begin, let us see exactly how much the potential drops at each step:

Lemma 6

$$\Psi_t - \Psi_{t+1} \geq \alpha^2/4$$

Proof Similar to before (with a couple of complicating details – query evaluation on the data structure must be checked to behave correctly). ■

Theorem 4 then immediately follows by combining Claim 5 with Lemma 6:

$$-\frac{1}{e} \leq \Psi_T \leq \log s - L \cdot \frac{\alpha^2}{4}$$

Solving for L we find:

$$L \leq 4 \frac{\log s + 1/e}{\alpha^2} < 4 \frac{\log s + 1}{\alpha^2}$$

Finally to see that the SMW data structure never reports **FAILURE**, it suffices to observe that $\text{ind}_T \leq s$. Because each query Q_t is assumed to be m-sparse, at most m variables can be **add**ed to the SMW data structure at each update. Therefore, we have

$$\text{ind}_T \leq m \cdot T \leq \frac{4m(\log s + 1)}{\alpha^2} \leq s$$

The last inequality follows from recalling that we chose s such that $s/(\log s + 1) \geq 4m/\alpha^2$. This completes the proof. ■

As a result, we get:

Theorem 7 *For any $0 < \epsilon, \delta, \beta < 1$ There exists an (ϵ, δ)-differentially private query release mechanism in the interactive setting, with running time per query $\tilde{O}(m/\alpha^2)$ that is (α, β)-accurate with respect to the set of all m-sparse linear queries $\mathcal{Q}_m$, with:*

$$\alpha = O\left(\frac{(\log m)^{1/4} \left(\log \frac{4}{\delta} \log \frac{k}{\beta} \right)^{1/2}}{(\epsilon \cdot n)^{1/2}} \right)$$

Note that this theorem not only improves over our general results in terms of running time (running time m/α^2 instead of $|X|$), it improves on them in terms of the accuracy it achieves for sparse queries! Where $\log |X|$ previously appeared, now appears only $\log m$!

This means that we can handle queries even over an infinite universe (i.e. with string-valued elements with no a-priori upper bound on string length). Of course our algorithm must be able to read the name of each universe element encountered, and so the running time of the algorithm will have some dependence on the length of the maximum length string encountered during the run of the algorithm, but no dependence on any a-priori (unboundedly large!) upper bound on string length. The accuracy will continue to have *no* dependence on universe size!

Bibliographic Information The sparse multiplicative weights algorithm is from Blum and Roth, "Fast Private Data Release Algorithms for Sparse Queries", 2011. It is a modification of the multiplicative weights algorithm of Hardt and Rothblum, given in "A Multiplicative Weights Mechanism for Privacy Preserving Data Analysis", 2010. The modification is analogous to the modification of the online learning algorithm Winnow to work in the infinite attribute model, given by Blum in "Learning Boolean Functions in an Infinite Attribute Space", 1990.