A Simple and Practical Algorithm for Differentially Private Data Release

Moritz Hardt
IBM Almaden Research
San Jose, CA
mhardt@us.ibm.com

Katrina Ligett*
Caltech
katrina@caltech.edu

Frank McSherry
Microsoft Research SVC
mcsherry@microsoft.com

Abstract

We present a new algorithm for differentially private data release, based on a simple combination of the Exponential Mechanism with the Multiplicative Weights update rule. Our *MWEM* algorithm achieves what are the best known and nearly optimal theoretical guarantees, while at the same time being simple to implement and experimentally more accurate on actual data sets than existing techniques.

1 Introduction

Sensitive statistical data on individuals are ubiquitous, and publishable analysis of such private data is an important objective. When releasing statistics or synthetic data based on sensitive data sets, one must balance the inherent tradeoff between the usefulness of the released information and the privacy of the affected individuals. Against this backdrop, differential privacy [1, 2, 3] has emerged as a compelling privacy definition that allows one to understand this tradeoff via formal, provable guarantees. In recent years, the theoretical literature on differential privacy has provided a large repertoire of techniques for achieving the definition in a variety of settings (see, e.g., [4, 5]). However, data analysts have found that several algorithms for achieving differential privacy add unacceptable levels of noise.

In this work we develop a broadly applicable, simple, and easy-to-implement algorithm, capable of substantially improving the performance of *linear queries* on many realistic datasets. Linear queries are equivalent to statistical queries (in the sense of [6]) and can serve as the basis of a wide range of data analysis and learning algorithms (see [7] for some examples).

Our algorithm is a combination of the Multiplicative Weights approach of [8, 9], maintaining and correcting an approximating distribution through queries on which the approximate and true datasets differ, and the Exponential Mechanism [10], which selects the queries most informative to the Multiplicative Weights algorithm (specifically, those most incorrect vis-a-vis the current approximation). One can view our approach as combining expert learning techniques (multiplicative weights) with an active learning component (via the exponential mechanism).

We present experimental results for producing differentially private synthetic data for a variety of problems studied in prior work: range queries as studied by [11, 12], contingency table release across a collection of statistical benchmarks as in [13], and datacube release as studied by [14]. We empirically evaluate the accuracy of the differentially private data produced by MWEM using the same query class and accuracy metric proposed by the corresponding prior work, improving on all. Beyond empirical improvements in these settings, MWEM matches the best known and nearly optimal theoretical accuracy guarantees for differentially private data analysis with linear queries.

*Computer Science Department, Cornell University. Work supported in part by an NSF Computing Innovation Fellowship (NSF Award CNF-0937060) and an NSF Mathematical Sciences Postdoctoral Fellowship (NSF Award DMS-1004416).

Finally, we describe a scalable implementation of MWEM capable of processing and releasing datasets of substantial complexity. Producing synthetic data for the classes of queries we consider is known to be computationally hard in the *worst-case* [15, 16]. Indeed, almost all prior work performs computation proportional to the size of the data domain, which limits them to datasets with relatively few attributes. In contrast, we are able to process datasets with thousands of attributes, corresponding to domains of size 2^{1000}. Our implementation integrates a scalable parallel implementation of Multiplicative Weights, and a representation of the approximating dataset in a factored form that only exhibits complexity when the model requires it.

2 Our Approach

The MWEM algorithm (Figure 1) maintains an approximating dataset as a (scaled) distribution over the domain D of data records. We repeatedly improve the accuracy of this approximation with respect to the private dataset and the desired query set by selecting and posing a query poorly served by our approximation and improving the approximation to better reflect the answer to this query. We select and pose queries using the Exponential [10] and Laplace Mechanisms [3], whose definitions and privacy properties we review in Subsection 2.1. We improve our approximation using the Multiplicative Weights update rule [8], reviewed in Subsection 2.2.

2.1 Differential Privacy and Mechanisms

Differential privacy is a constraint on a randomized computation that the computation should not reveal specifics of individual records present in the input. It places this constraint by requiring the mechanism to behave almost identically on any two datasets that are sufficiently close.

Imagine a dataset A whose records are drawn from some abstract domain D, and which is described as a function from D to the natural numbers $\mathbb{N}$, with $A(x)$ indicating the frequency (number of occurrences) of x in the dataset. We use $\|A - B\|$ to indicate the sum of the absolute values of difference in frequencies (how many records would have to be added or removed to change A to B).

Definition 2.1 (Differential Privacy)**.** *A mechanism M mapping datasets to distributions over an output space R provides (ε, δ)-differential privacy if for every $S \subseteq R$ and for all data sets A, B where $\|A - B\| \leq 1$, $\Pr[M(A) \in S] \leq e^{\varepsilon} \Pr[M(B) \in S] + \delta$. If $\delta = 0$ we say that M provides ε-differential privacy.*

The Exponential Mechanism [10] is an ε-differentially private mechanism that can be used to select among the best of a discrete set of alternatives, where "best" is defined by a function relating each alternative to the underlying secret data. Formally, for a set of alternative results R, we require a quality scoring function $s : dataset \times R \to \mathbb{R}$, where $s(B, r)$ is interpreted as the quality of the result r for the dataset B. To guarantee ε-differential privacy, the quality function is required to satisfy a stability property: that for each result r the difference $|s(A, r) - s(B, r)|$ is at most $\|A - B\|$. The Exponential Mechanism E simply selects a result r from the distribution satisfying

$$\Pr[E(B) = r] \propto \exp(\varepsilon \times s(B, r)/2).$$

Intuitively, the mechanism selects result r biased exponentially by its quality score. The Exponential Mechanism takes time linear in the number of possible results, evaluating $s(B, r)$ once for each r.

A *linear query* (also referred to as *counting query* or *statistical query*) is specified by a function q mapping data records to the interval $[-1, +1]$. The answer of a linear query on a data set D, denoted $q(B)$, is the sum $\sum_{x \in D} q(x) \times B(x)$.

The Laplace Mechanism is an ε-differentially private mechanism which reports approximate sums of bounded functions across a dataset. If q is a linear query, the Laplace Mechanism L obeys

$$\Pr[L(B) = r] \propto \exp\left(-\varepsilon \times |r - q(B)|\right)$$

Although the Laplace Mechanism is an instance of the Exponential Mechanism, it can be implemented much more efficiently, by adding Laplace noise with parameter $1/\varepsilon$ to the value $q(B)$. As the Laplace distribution is exponentially concentrated, the Laplace Mechanism provides an excellent approximation to the true sum.

Inputs: Data set B over a universe D, set Q of linear queries; Number of iterations $T \in \mathbb{N}$; Privacy parameter $\varepsilon > 0$.

Let n denote $\|B\|$, the number of records in B. Let A_0 denote n times the uniform distribution over D. For iteration $i = 1, ..., T$:

1. *Exponential Mechanism:* Sample a query $q_i \in Q$ using the Exponential Mechanism parametrized with epsilon value $\varepsilon/2T$ and the score function
$$s_i(B, q) = |q(A_{i-1}) - q(B)| \ .$$
2. *Laplace Mechanism:* Let measurement $m_i = q_i(B) + Lap(2T/\varepsilon)$.
3. *Multiplicative Weights:* Let A_i be n times the distribution whose entries satisfy
$$A_i(x) \propto A_{i-1}(x) \times \exp(q_i(x) \times (m_i - q_i(A_{i-1}))/2n) \ .$$

Output: $A = \text{avg}_{i<T} A_i$.

Figure 1: The MWEM algorithm.

2.2 Multiplicative Weights Update Rule

The Multiplicative Weights approach has seen application in many areas of computer science. Here we will use it as proposed in Hardt and Rothblum [8], to repeatedly improve an approximate distribution to better reflect some true distribution. The intuition behind Multiplicative Weights is that should we find a query whose answer on the true data is much larger than its answer or the approximate data, we should scale up the approximating weights on records contributing positively and scale down the weights on records contributing negatively. If the true answer is much less than the approximate answer, we should do the opposite.

More formally, let q be a linear query. If A and B are distributions over the domain D of records, where A is a synthetic distribution intended to approximate a true distribution B with respect to query q, then the Multiplicative Weights update rule recommends updating the weight A places on each record x by:
$$A_{new}(x) \propto A(x) \times \exp(q(x) \times (q(B) - q(A))/2) \ .$$
The proportionality sign indicates that the approximation should be renormalized after scaling. Hardt and Rothblum show that each time this rule is applied, the relative entropy between A and B decreases by an additive $(q(A) - q(B))^2$. As long as we can continue to find queries on which the two disagree, we can continue to improve the approximation.

Although our algorithm will manipulate datasets, we can divide their frequencies by the numbers of records n whenever we need to apply Multiplicative Weights updates, and renormalize to a fixed number of records (rather than to one, as we would for a distribution).

2.3 Formal Guarantees

As indicated in the introduction, the formal guarantees of MWEM represent the best known theoretical results on differentially private synthetic data release. We first describe the privacy properties.

Theorem 2.1. *MWEM satisfies ε-differential privacy.*

Proof. The composition rules for differential privacy state that ε values accumulate additively. We make T calls to the Exponential Mechanism with parameter $(\varepsilon/2T)$ and T calls to the Laplace Mechanism with parameter $(\varepsilon/2T)$, resulting in ε-differential privacy. □

We now bound the worst-case performance of the algorithm, in terms of the maximum error between A and B across all $q \in Q$. The natural range for $q(A)$ is $[-n, +n]$, and we see that by increasing T beyond $4 \log |D|$ we can bring the error asymptotically smaller than n.

Theorem 2.2. *For any dataset B, set of linear queries Q, $T \in \mathbb{N}$, and $\varepsilon > 0$, with probability at least $1 - 2T/|Q|$, MWEM produces A such that*

$$\max_{q \in Q} |q(A) - q(B)| \leq 2n\sqrt{\frac{\log |D|}{T}} + \frac{10T \log |Q|}{\varepsilon} .$$

Proof. The proof of this theorem is an integration of pre-existing analyses of both the Exponential Mechanism and the Multiplicative Weights update rule, omitted for reasons of space. □

Note that these bounds are worst-case bounds, over adversarially chosen data and query sets. We will see in Section 3 that MWEM works very well in more realistic settings.

2.3.1 Running time

The running time of our basic algorithm as described in Figure 1 is $O(n|Q| + T|D||Q|))$. The algorithm is embarrassingly parallel: query evaluation can be conducted independently, implemented using modern database technology; the only required serialization is that the T steps must proceed in sequence, but within each step essentially all work is parallelizable.

Results of Dwork et al. [17] show that for worst case data, producing differentially private synthetic data for a set of counting queries requires time $|D|^{0.99}$ under reasonable cryptographic hardness assumptions. Moreover, Ullman and Vadhan [16] showed that similar lower bounds also hold for more basic query classes such as we consider in Section 3.2. Despite these hardness results, we provide an alternate implementation of our algorithm in Section 4 and demonstrate that its running time is acceptable on real-world data even in cases where $|D|$ is as large as 2^{77}, and on simple synthetic input datasets where $|D|$ is as large as 2^{1000}.

2.3.2 Improvements and Variations

There are several ways to improve the empirical performance of MWEM at the expense of the theoretical guarantees. First, rather than use the average of the distributions A_i we use only the final distribution. Second, in each iteration we apply the multiplicative weights update rule for all measuments taken, multiple times; as long as any measurements do not agree with the approximating distribution (within error) we can improve the result. Finally, it is occasionally helpful to initialize A_0 by performing a noisy count for each element of the domain; this consumes from the privacy budget and lesens the accuracy of subsequent queries, but is often a good trade-off.

2.4 Related Work

The study of differentially private synthetic data release mechanisms for arbitrary counting queries began with the work of Blum, Ligett, and Roth [18], who gave a computationally inefficient (superpolynomial in $|D|$) ε-differentially private algorithm that achieves error that scales only logarithmically with the number of queries. The dependence on n and $|Q|$ achieved by their algorithm is $O(n^{2/3} \log^{1/3} |Q|)$ (which is the same dependence achieved by optimizing the choice of T in Theorem 2.2). Since [18], subsequent work [17, 19, 20, 8] has focused on computationally more efficient algorithms (i.e., polynomial in $|D|$) as well as algorithms that work in the interactive query setting. The latest of these results is the private Multiplicative Weights method of Hardt and Rothblum [8] which achieves error rates of $O(\sqrt{n \log(|Q|)})$ for (ε, δ)-differential privacy (which is the same dependence achieved by applying *k-fold adaptive composition* [19] and optimizing T in our Theorem 2.2). While their algorithm works in the interactive setting, it can also be used non-interactively to produce synthetic data, albeit at a computational overhead of $O(n)$. MWEM can also be cast as an instance of a more general Multiplicative-Weights based framework of Gupta et al. [9], though our specific instantiation and its practical appeal were not anticipated in their work.

Prior work on linear queries includes Fienberg et al. [13] and Barak et al. [21] on contingency tables; Li et al. [22] on range queries (and substantial related work [23, 24, 22, 11, 12, 25] which Li and Miklau [11, 25] show can all be seen as instances of the matrix mechanism of [22]); and Ding et al. [14] on data cubes. In each case, MWEM's theoretical guarantees and experimental performance improve on prior work. We compare further in Section 3.

3 Experimental Evaluation

We evaluate MWEM across a variety of query classes, datasets, and metrics as explored by prior work, demonstrating improvement in the quality of approximation (often significant) in each case. The problems we consider are: (1) range queries under the total squared error metric, (2) binary contingency table release under the relative entropy metric, and (3) datacube release under the average absolute error metric. Although contingency table release and datacube release are very similar, prior work on the two have had different focuses: small datasets over many binary attributes vs. large datasets over few categorical attributes, low-order marginals vs. all cuboids as queries, and relative entropy vs. the average error within a cuboid as metrics.

Our general conclusion is that intelligently selecting the queries to measure can result in significant accuracy improvements, in settings where accuracy is a scare resource. When the privacy parameters are very lax, or the query set very simple, direct measurement of all queries yields better results than expending some fraction of the privacy budget determining what to measure. On the other hand, in the more challenging case of restrictions on privacy for complex data and query sets, MWEM can substantially out-perform previous algorithms.

3.1 Range Queries

A range query over a domain $D = \{1, \ldots, N\}$ is a counting query specified by the indicator function of an interval $I \subseteq D$. Over a multi-dimensional domain $D = D_1 \times \ldots D_d$ a range query is defined by the product of indicator functions. Differentially private algorithms for range queries were specifically considered by [18, 23, 24, 22, 11, 12, 25]. As noted in [11, 25], all previously implemented algorithms for range queries can be seen as instances of the *matrix mechanism* of [22]. Moreover, [11, 25] show a *lower bound* on the total squared error achieved by the matrix mechanism in terms of the singular values of a matrix associated with the set of queries. We refer to this bound as the *SVD bound*.

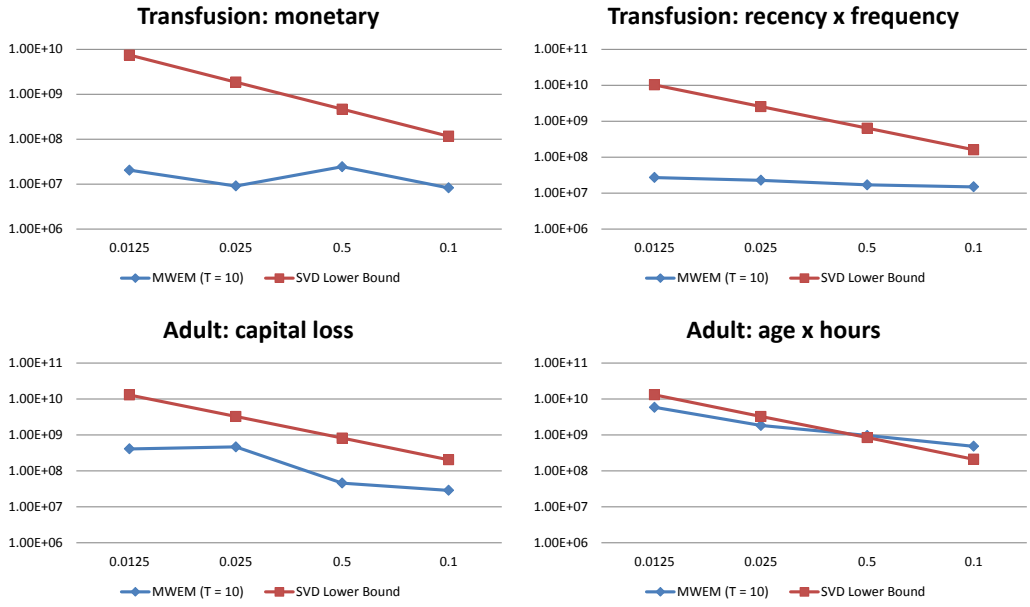

Figure 2: Comparison of MWEM with the SVD lower bound on four data sets. The y-axis measures the average squared error per query, averaged over 5 independent repetitions of the experiment, as epsilon varies. The improvement is most significant for small epsilon, diminishing as epsilon increases.

We empirically evaluate MWEM for range queries on restrictions of the Adult data set [26] to (a) "capital loss" attribute, and (b) the "age" and "hours" attributes, as well as the restriction of the

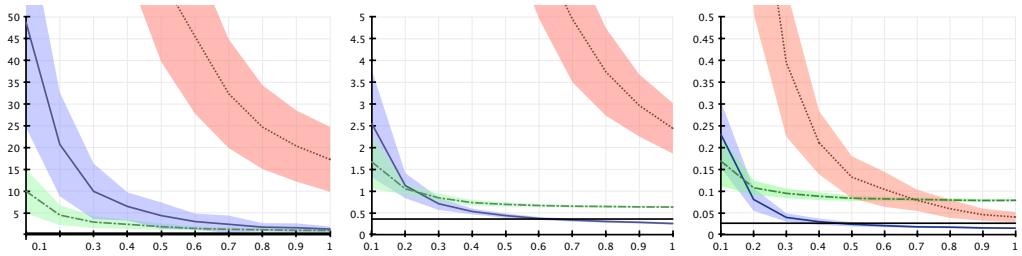

Figure 3: Relative entropy (y-axis) as a function of epsilon (x-axis) for the mildew, rochdale, and czech datasets, respectively. The lines represent averages across 100 runs, and the corresponding shaded areas one standard deviation in each direction. Red (dashed) represents the modified Barak et al. [21] algorithm, green (dot-dashed) represents unoptimized MWEM, and blue (solid) represents the optimized version thereof. The solid black horizontal line is the stated relative entropy values from Fienberg et al. [13].

Blood Transfusion data set [26, 27] to (c) the "recency" and "frequency" attributes, and (d) the "monetary" attribute. We chose these data sets as they feature numerical attributes of suitable size. In Figure 2, we compare the performance of MWEM on sets of randomly chosen range queries against the SVD lower bound proved by [11, 25], varying ε while keeping the number of queries fixed. The SVD lower bound holds for algorithms achieving the strictly weaker guarantee of (ε, δ)-differential privacy with $\delta > 0$, permitting some probability δ of unbounded disclosure. The SVD bound depends on δ; in our experiments we fixed $\delta = 1/n$ when instantiating the SVD bound, as any larger value of δ permits mechanisms capable of exact release of individual records.

3.2 Contingency Tables

A contingency table can be thought of as a table of records over d binary attributes, and the k-way marginals of a contingency table correspond to the $\binom{d}{k}$ possible choices of k attributes, where each marginal is represented by the 2^k counts of the records with each possible setting of attributes. In previous work, Barak et al. [21] describe an approach to differentially private contingency table release using linear queries defined by the Hadamard matrix. Importantly, all k-dimensional marginals can be exactly recovered by examination of relatively few such queries: roughly $\binom{d}{k}$ out of the possible 2^d, improving over direct measurement of the marginals by a factor of 2^k. This algorithm is evaluated by Fienberg et al. [13], and was found to do poorly on several benchmark datasets.

We evaluate our approximate dataset following Fienberg et al. [13] using *relative entropy*, also known as the Kullback-Leibler (or KL) divergence. Formally, the relative entropy between our two distributions (A/n and B/n) is

$$RE(B||A) = \sum_{x \in D} B(x) \log(B(x)/A(x))/n \ .$$

We use several statistical datasets from Fienberg et al. [13], and evaluate two variants of MWEM (both with and without initialization of A_0) against a modification of Barak et al. [21] which combines its observations using multiplicative weights (we find that without this modification, [21] is terrible with respect to relative entropy). These experiments are therefore largely assessing the selective choice of measurements to take, rather than the efficacy of multiplicative weights.

Figure 3 presents the evaluation of MWEM on several small datasets in common use by statisticians. Our findings here are fairly uniform across the datasets: the ability to measure only those queries that are informative about the dataset results in substantial savings over taking all possible measurements. In many cases MWEM approaches the good non-private values of [13], indicating that we can approach levels of accuracy at the limit of statistical validity.

We also consider a larger dataset, the National Long-Term Care Study (NLTCS), in Figure 4. This dataset contains orders of magnitudes more records, and has 16 binary attributes. For our initial settings, maintaining all three-way marginals, we see similar behavior as above: the ability to choose

the measurements that are important allows substantially higher accuracy on those that matter. However, we see that the algorithm of Barak et al. [21] is substantially more competitive in the regime where we are interested in querying all two-dimensional marginals, rather than the default three we have been using. In this case, for values of epsilon at least 0.1, it seems that there is enough signal present to simply measure all corresponding entries of the Hadamard transform; each is sufficiently informative that measuring substantially fewer at higher accuracy imparts less information, rather than more.

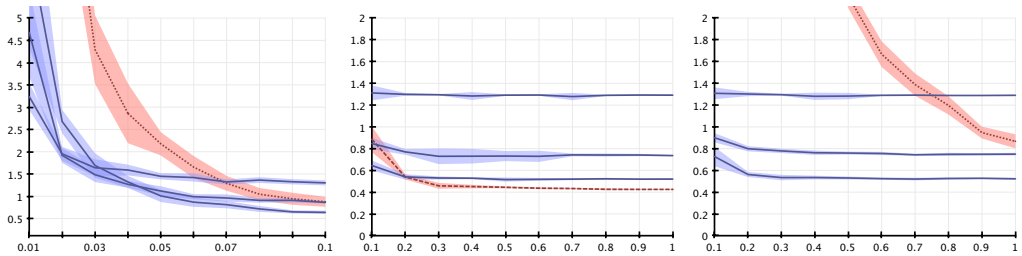

Figure 4: Curves comparing our approach with that of Barak et al. on the National Long Term Care Survey. The red (dashed) curve represents Barak et al, and the multiple blue (solid) curves represent MWEM, with 20, 30, and 40 queries (top to bottom, respectively). From left to right, the first two figures correspond to degree 2 marginals, and the third to degree 3 marginals. As before, the x-axis is the value of epsilon guaranteed, and the y-axis is the relative entropy between the produced distribution and actual dataset. The lines represent averages across only 10 runs, owing to the high complexity of Barak et al. on this many-attributed dataset, and the corresponding shaded areas one standard deviation in each direction.

3.3 Data Cubes

We now change our terminology and objectives, shifting our view of contingency tables to one of datacubes. The two concepts are interchangeable, a contingency table corresponding to the datacube, and a marginal corresponding to its cuboids. However, the datasets studied and the metrics applied are different. We focus on the restriction of the Adult dataset [26] to its eight categorical attributes, as done in [14], and evaluate our approximations using average error within a cuboid, also as in [14].

Although MWEM is defined with respect to a single query at a time, it generalizes to sets of counting queries, as reflected in a cuboid. The Exponential Mechanism can select a cuboid to measure using a quality score function summing the absolute values of the errors within the cells of the cuboid. We also (heuristically) subtract the number of cells from the score of a cuboid to bias the selection away from cuboids with many cells, which would collect Laplace error in each cell. This subtraction does not affect privacy properties. An entire cuboid can be measured with a single differentially private query, as any record contributes to at most one cell (this is a generalization of the Laplace Mechanism to multiple dimensions, from [3]). Finally, Multiplicative Weights works unmodified, increasing and decreasing weights based on the over- or under-estimation of the count to which the record contributes.

We compare MWEM with the work of [14] in Figure 5. The average average error improves noticeably, by approximately a factor of four. The maximum average error is less clear; experimentally we have found we can bring the numbers lower using different heuristic variants of MWEM, but without principled guidance we report only the default behavior. Of note, our results are achieved by a single algorithm, whereas the best results for maximum and average error in [14] are achieved by different algorithms, each designed to optimize one specific metric.

4 A Scalable Implementation

The implementation of MWEM used in the previous experiments quite literally maintains a distribution A_i over the elements of the universe D. As the number of attributes grows, the universe D grows exponentially, and it can quickly become infeasible to track the distribution explicitly. In

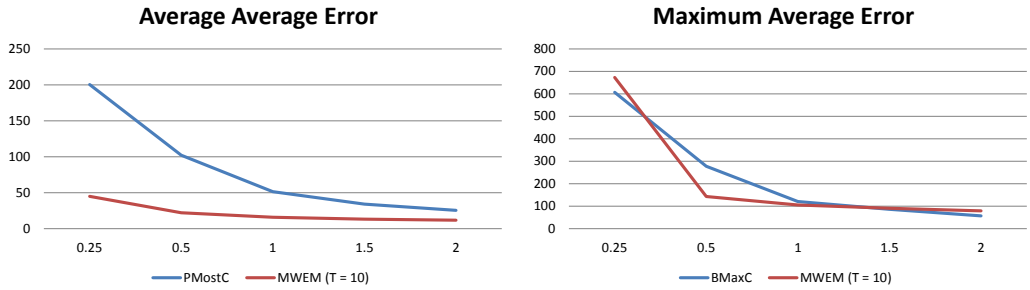

Figure 5: Comparison of MWEM with the custom approaches from [14], varying epsilon through the reported values from [14]. Each cuboid (marginal) is assessed by its average error, and either the average or maximum over all 256 marginals is taken to evaluate the technique.

this section, we consider a scalable implementation with essentially no memory footprint, whose running time is in the worst case proportional to $|D|$, but which for many classes of simple datasets remains linear in the number of attributes.

Recall that the heart of MWEM maintains a distribution A_i over D that is then used in the Exponential Mechanism to select queries poorly approximated by the current distribution. From the definition of the Multiplicative Weights distribution, we see that the weight $A_i(x)$ can be determined from the history $H_i = \{(q_j, m_j) : j \leq i\}$:

$$A_i(x) \propto \exp\left(\sum_{j \leq i} q_j(x) \times (m_j - q_j(A_{j-1}))/2n\right) .$$

We explicitly record the scaling factors $l_j = m_j - q_j(A_{j-1})$ as part of the history $H_i = \{(q_j, m_j, l_j) : j \leq i\}$, to remove the dependence on prior A_j.

The domain D is often the product of many attributes. If we partition these attributes into disjoint parts $D_1, D_2, \ldots D_k$ so that no query in H_i involves attributes from more than one part, then the distribution produced by Multiplicative Weights is a product distribution over $D_1 \times D_2 \times \ldots D_k$. For query classes that factorize over the attributes of the domain (for example, range queries, marginal queries, and cuboid queries) we can rewrite and efficiently perform the integration over D using

$$\sum_{x \in D_1 \times D_2 \times \ldots D_k} q(x) \times A_i(x) = \prod_{1 \leq j \leq k} \left(\sum_{x_j \in D_j} q(x_j) \times A_i^j(x_j)\right) .$$

where A_i^j is a mini Multiplicative Weights over attributes in part D_j, using only the relevant queries from H_i. So long as the measurements taken reflect modest groups of independent attributes, the integration can be efficiently performed. As the measurements overlap more and more, additional computation or approximation is required. The memory footprint is only the combined size of the data, query, and history sets.

Experimentally, we are able to process a binarized form of the Adult dataset with 27 attributes efficiently (taking 80 seconds to process completely), and the addition of 50 new independent binary attributes, corresponding to a domain of size 2^{77}, results in neglible performance impact. For a simple synthetic dataset with up to 1,000 independent binary attributes, the factorized implementation of MWEM takes only 19 seconds to for a complete execution.

5 Conclusions

We introduced MWEM, a simple algorithm for releasing data maintaining a high fidelity to the protected source data, as well as differential privacy with respect to the records. The approach builds upon the Multiplicative Weights approach of [8, 9], by introducing the Exponential Mechanism [10]

as a more judicious approach to determining which measurements to take. The theoretical analysis matches previous work in the area, and experimentally we have evidence that for many interesting settings, MWEM represents a substantial improvement over existing techniques.

As well as improving on experimental error, the algorithm is both simple to implement and simple to use. An analyst does not require a complicated mathematical understanding of the nature of the queries (as the community has for linear algebra [11] and the Hadamard transform [21]), but rather only needs to enumerate those measurements that should be preserved. We hope that this generality leads to a broader class of high fidelity differentially-private data releases across a variety of data domains.

References

[1] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, 2003.

[2] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*. Springer, 2004.

[3] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[4] Cynthia Dwork. The differential privacy frontier (extended abstract). In *TCC*, 2009.

[5] Cynthia Dwork. The promise of differential privacy: A tutorial on algorithmic techniques. In *FOCS*, 2011.

[6] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

[7] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In *Proc. 24th PODS*, pages 128–138. ACM, 2005.

[8] Moritz Hardt and Guy Rothblum. A multiplicative weights mechanism for interactive privacy-preserving data analysis. In *FOCS*, 2010.

[9] Anupam Gupta, Moritz Hardt, Aaron Roth, and Jon Ullman. Privately releasing conjunctions and the statistical query barrier. In *STOC*, 2011.

[10] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, 2007.

[11] Chao Li and Gerome Miklau. Efficient batch query answering under differential privacy. *CoRR*, abs/1103.1367, 2011.

[12] Chao Li and Gerome Miklau. An adaptive mechanism for accurate query answering under differential privacy. *to appear, PVLDB*, 2012.

[13] Stephen E. Fienberg, Alessandro Rinaldo, and Xiolin Yang. Differential privacy and the risk-utility tradeoff for multi-dimensional contingency tables. In *Privacy in Statistical Databases*, 2010.

[14] Bolin Ding, Marianne Winslett, Jiawei Han, and Zhenhui Li. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD*, 2011.

[15] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, 2009.

[16] Jonathan Ullman and Salil P. Vadhan. PCPs and the hardness of generating private synthetic data. In *TCC*, 2011.

[17] C. Dwork, M. Naor, O. Reingold, G.N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, 2009.

[18] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *STOC*, 2008.

[19] Cynthia Dwork, Guy Rothblum, and Salil Vadhan. Boosting and differential privacy. In *FOCS*, 2010.

[20] Aaron Roth and Tim Roughgarden. The median mechanism: Interactive and efficient privacy with multiple queries. In *STOC*, 2010.

[21] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS*, 2007.

[22] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, 2010.

[23] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on Knowledge and Data Engineering*, 23:1200–1214, 2011.

[24] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially-private queries through consistency. In *VLDB*, 2010.

[25] Chao Li and Gerome Miklau. Measuring the achievable error of query sets under differential privacy. *CoRR*, abs/1202.3399v2, 2012.

[26] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[27] I-Cheng Yeh, King-Jang Yang, and Tao-Ming Ting. Knowledge discovery on RFM model using Bernoulli sequence. *Expert Systems with Applications*, 36(3), 2008.

A Appendix: Proof of Theorem 2.2

The proof of Theorem 2.2 is broken into two parts. First, we argue that across all T iterations, the queries q_i selected by the Exponential Mechanism are nearly optimal, and the errors introduced into m_i by the Laplace Mechanism are small. We then apply the potential function analysis of Hardt and Rothblum [8] to show that the maximum approximation error for any query cannot be too large.

Using the shorthand $\text{maxerr}_i \stackrel{\text{def}}{=} \max_j |q_j(A_{i-1}) - q_j(B)|$ and $\text{adderr} \stackrel{\text{def}}{=} 2T \log |Q|/\varepsilon$, we first claim that with high probability the Exponential Mechanism and Laplace Mechanism give nearly optimal results.

Lemma A.1. *With probability at least $1 - 2T/|Q|^c$, for any $c \geq 0$, for all $1 \leq i \leq T$, we have that both*

$$\begin{aligned} |q_i(A_{i-1}) - q_i(B)| &\geq \text{maxerr}_i - (2c+2) \times \text{adderr} \\ |m_i - q_i(B)| &\leq c \times \text{adderr} \ . \end{aligned}$$

Proof. The probability the Exponential Mechanism with parameter $\varepsilon/2T$ selects a query with quality score at least r less than the optimal (meaning the query on which A_{i-1} disagrees the most with B) is bounded by

$$\Pr[|q_i(A_{i-1}) - q_i(B)| < \text{maxerr}_i - r] \leq |Q| \times \exp(-\varepsilon r/4T).$$

If we take $r = (2c+2) \times 2T \log |Q|/\varepsilon$ the probability is at most $1/|Q|^c$ for each iteration.

By the definition of the Laplace distribution,

$$\Pr[|\text{Laplace}(2T/\varepsilon)| > r] \leq \exp(-r \times \varepsilon/2T) \ .$$

Taking $r = c \times 2T \log |Q|/\varepsilon$ bounds the probability by at most $1/|Q|^c$ for each iteration.

Taking a union bound over the $2T$ events, we arrive at a failure probability of at most $2T/|Q|^c$. □

We next argue that MWEM improves its approximation in each round where $q_i(A) - q_i(B)$ has large magnitude. To capture the improvement, we use the relative entropy again:

$$\Psi_i = \sum_{x \in D} B(x) \log(B(x)/A_i(x))/n \ .$$

The following two properties follow from non-negativity of entropy, and Jensen's Inequality:

Fact A.2. $\Psi_i \geq 0$

Fact A.3. $\Psi_0 \leq \log |D|$

We now show that the relative entropy decreases in each round by an amount reflecting the error q_i exposes between A_{i-1} and B, less the error in our measurement of $q_i(B)$.

Lemma A.4. *For each round $i \leq T$,*

$$\Psi_{i-1} - \Psi_i \geq \left(\frac{q_i(A_{i-1}) - q_i(B)}{2n}\right)^2 - \left(\frac{m_i - q_i(B)}{2n}\right)^2 \ .$$

Proof. We start by noting that

$$\Psi_{i-1} - \Psi_i = \sum_{x \in D} B(x) \log \left(\frac{A_i(x)}{A_{i-1}(x)} \right) / n \ .$$

The ratio $A_i(x)/A_{i-1}(x)$ can be written as $\exp(q_i(x)\eta_i)/\beta_i$, where $\eta_i = (m_i - q_i(A_{i-1}))/2n$ and β_i is the factor required to renormalize in round i. Using this notation,

$$\Psi_{i-1} - \Psi_i \quad = \quad \eta_i q_i(B)/n - \log \beta_i \ .$$

The required renormalization β_i equals

$$\beta_i = \sum_{x \in D} \exp(q_i(x)\eta_i) A_{i-1}(x)/n \ .$$

Using $\exp(x) \leq 1 + x + x^2$ for $|x| \leq 1$, and that $|q_i(x)\eta_i| \leq 1$,

$$\beta_i \leq \sum_{x \in D} (1 + q_i(x)\eta_i + q_i(x)^2 \eta_i^2) A_{i-1}(x)/n \ .$$

As $q_i(x)^2 \leq 1$, by assumption on all $q_i \in Q$, we have

$$\begin{aligned} \beta_i &\leq \sum_{x \in D} (1 + q_i(x)\eta_i + \eta_i^2) A_{i-1}(x)/n \\ &= 1 + \eta_i q_i(A_{i-1})/n + \eta_i^2 \ . \end{aligned}$$

Introducing this bound on β_i into our equality for $\Psi_{i-1} - \Psi_i$, and using $\log(1+x) \leq x$, we get

$$\Psi_{i-1} - \Psi_i \geq \eta_i (q_i(B) - q_i(A_{i-1}))/n - \eta_i^2 \ .$$

After reintroducing the definition of η_i and simplifying, this bound results in the statement of the lemma. □

With these two lemmas we are now prepared to prove Theorem 2.2, bounding the maximum error $|q(A) - q(B)|$.

(of Theorem 2.2). We start by noting that the quantity of interest, the maximum over queries q of the error between $q(A)$ and $q(B)$, can be rewritten and bounded by:

$$\begin{aligned} \max_{q \in Q} |q(A) - q(B)| &= \max_{q \in Q} |q(\operatorname*{avg}_{i \leq T} A_i) - q(B)| \\ &\leq \max_{q \in Q} \operatorname*{avg}_{i \leq T} |q(A_i) - q(B)| \\ &\leq \operatorname*{avg}_{i \leq T} \mathrm{maxerr}_i \ . \end{aligned}$$

At this point we invoke Lemma A.1 with $c = 1$ so that with probability at least $1 - 2T/|Q|$ we have for $i \leq T$ both

$$\begin{aligned} \mathrm{maxerr}_i &\leq |q_i(A_{i-1}) - q_i(B)| + 4 \times \mathrm{adderr} \ , \\ |m_i - q_i(B)| &\leq \mathrm{adderr} \ . \end{aligned}$$

Combining these bounds with those of Lemma A.4 gives

$$\mathrm{maxerr}_i \quad \leq \quad \left(4n^2(\Psi_{i-1} - \Psi_i) + \mathrm{adderr}^2\right)^{1/2} + 4 \times \mathrm{adderr} \ .$$

We now average over $i \leq T$, and apply Cauchy-Schwarz, specifically that $\operatorname{avg}_i x_i^{1/2} \leq (\operatorname{avg}_i x_i)^{1/2}$, giving

$$\begin{aligned} \operatorname*{avg}_{i \leq T} \mathrm{maxerr}_i \quad \leq \quad & \left(4n^2 \operatorname*{avg}_i (\Psi_{i-1} - \Psi_i) + \mathrm{adderr}^2\right)^{1/2} \\ & + 4 \times \mathrm{adderr} \ . \end{aligned}$$

The average $\text{avg}_i(\Psi_{i-1} - \Psi_i)$ telescopes to $(\Psi_0 - \Psi_T)/T$, which Facts A.2 and A.3 bound by $\log(|D|)/T$, giving

$$\operatorname*{avg}_{i \le T} \text{maxerr}_i \le \left(4n^2 \log(|D|)/T + \text{adderr}^2\right)^{1/2} + \ 4 \times \text{adderr} \ .$$

Finally, as $\sqrt{a+b} \le \sqrt{a} + \sqrt{b}$, we derive

$$\operatorname*{avg}_{i \le T} \text{maxerr}_i \quad \le \quad 2n(\log(|D|)/T)^{1/2} + 5 \times \text{adderr} \ .$$

If we substitute $2T \log |Q|/\varepsilon$ for adderr we get the bound in the statement of the theorem. Replacing the factor of 5 by $(3c+2)$ generalizes the result to hold with probability $1 - 2T/|Q|^c$ for arbitrary $c > 0$. □

```
double[] MWEM(double[] B, Func<int, double>[] Q, int T, double eps)
{
    var n = (double) B.Sum();                                       // should be taken privately, we ignore
    var A = Enumerable.Repeat(n / B.Length, B.Length).ToArray();  // approx dataset, initially uniform
    var measurements = new Dictionary<int, double>();              // records (qi, mi) measurement pairs
    var random = new Random();                                      // RNG used all over the place

    for (int i = 0; i < T; i++)
    {
        // determine a new query to measure, rejecting prior queries
        var qi = random.ExponentialMechanism(B, A, Q, eps / (2 * T));
        while (measurements.ContainsKey(qi))
            qi = random.ExponentialMechanism(B, A, Q, eps / (2 * T));

        // measure the query, and add it to our collection of measurements
        measurements.Add(qi, Q[qi].Evaluate(B) + random.Laplace((2 * T) / eps));

        // improve the approximation using poorly fit measurements
        A.MultiplicativeWeights(Q, measurements);
    }

    return A;
}

int ExponentialMechanism(this Random random, double[] B, double[] A, Func<int, double>[] Q, double eps)
{
    var errors = new double[Q.Length];
    for (int i = 0; i < errors.Length; i++)
        errors[i] = eps * Math.Abs(Q[i].Evaluate(B) - Q[i].Evaluate(A)) / 2.0;

    var maximum = errors.Max();
    for (int i = 0; i < errors.Length; i++)
        errors[i] = Math.Exp(errors[i] - maximum);

    var uniform = errors.Sum() * random.NextDouble();
    for (int i = 0; i < errors.Length; i++)
    {
        uniform -= errors[i];
        if (uniform <= 0.0)
            return i;
    }

    return errors.Length - 1;
}

double Laplace(this Random random, double sigma)
{
    return sigma * Math.Log(random.NextDouble()) * (random.Next(2) == 0 ? -1 : +1);
}

void MultiplicativeWeights(this double[] A, Func<int, double>[] Q, Dictionary<int, double> measurements)
{
    var total = A.Sum();

    for (int iteration = 0; iteration < 100; iteration++)
    {
        foreach (var qi in measurements.Keys)
        {
            var error = measurements[qi] - Q[qi].Evaluate(A);
            for (int i = 0; i < A.Length; i++)
                A[i] *= Math.Exp(Q[qi](i) * error / (2.0 * total));

            var count = A.Sum();
            for (int i = 0; i < A.Length; i++)
                A[i] *= total / count;
        }
    }
}

double Evaluate(this Func<int, double> query, double[] collection)
{
    return Enumerable.Range(0, collection.Length).Sum(i => query(i) * collection[i]);
}
```

Figure 6: Full source code for a reference implementation of MWEM.