Iterative Methods for Private Synthetic Data: Unifying Framework and New Methods

Terrance Liu*
Carnegie Mellon University
Pittsburgh, PA 15213
terrancl@andrew.cmu.edu

Giuseppe Vietri*
University of Minnesota
Minneapolis, MN 55455
vietr002@umn.edu

Zhiwei Steven Wu
Carnegie Mellon University
Pittsburgh, PA 15213
zstevenwu@cmu.edu

Abstract

We study private synthetic data generation for query release, where the goal is to construct a sanitized version of a sensitive dataset, subject to differential privacy, that approximately preserves the answers to a large collection of statistical queries. We first present an algorithmic framework that unifies a long line of iterative algorithms in the literature. Under this framework, we propose two new methods. The first method, private entropy projection (PEP), can be viewed as an advanced variant of MWEM that adaptively reuses past query measurements to boost accuracy. Our second method, generative networks with the exponential mechanism (GEM), circumvents computational bottlenecks in algorithms such as MWEM and PEP by optimizing over generative models parameterized by neural networks, which capture a rich family of distributions while enabling fast gradient-based optimization. We demonstrate that PEP and GEM empirically outperform existing algorithms. Furthermore, we show that GEM nicely incorporates prior information from public data while overcoming limitations of PMW^{Pub}, the existing state-of-the-art method that also leverages public data.

1 Introduction

In this paper, we study the problem of differentially private query release: given a large collection of statistical queries, the goal is to release approximate answers subject to the constraint of differential privacy. We focus on the approach of synthetic data generation—that is, generate a privacy-preserving "fake" dataset, or more generally a representation of a probability distribution, that approximates all statistical queries of interest.

There has been a recent surge of work on practical algorithms for generating private synthetic data. Even though they differ substantially in details, these algorithms share the same iterative form that maintains and improves a probability distribution over the data domain: identifying a small collection of high-error queries each round and updating the distribution to reduce these errors. Inspired by this observation, we present a unifying algorithmic framework that captures these methods. Furthermore, we develop two new algorithms, PEP and GEM, and extend the latter to the setting in which public data is available. We summarize our contributions below:

Unifying algorithmic framework. We provide a framework that captures existing iterative algorithms and their variations. We then argue that under this framework, the optimization procedures of each method can be reduced to what loss function is minimized and how its distributional family is parameterized (see Appendix B.2).

Private Entropy Projection (PEP). The first algorithm we propose is PEP, which can be viewed as a more advanced version of MWEM with an adaptive and optimized learning rate. We show that

*First two authors contributed equally.

Preprint. Under review.

PEP minimizes a regularized exponential loss function that can be efficiently optimized using an iterative procedure. Moreover, we show that PEP monotonically decreases the error over rounds and empirically finds that it achieves higher accuracy and faster convergence than MWEM.

Generative networks with the exponential mechanism (GEM). Like MWEM, PEP explicitly maintains a joint distribution over the data domain, resulting in a runtime that is exponential in the dimension of the data. Our second method, GEM, avoids this fundamental issue by optimizing the absolute loss over a set of generative models parameterized by neural networks. We empirically demonstrate that in the high-dimensional regime, GEM outperforms all competing methods.

Incorporating public data. Finally, we consider extensions of our methods that incorporate prior information in publicly available datasets (e.g., previous data releases from the American Community Survey (ACS) prior to their differential privacy deployment). We then demonstrate empirically that GEM circumvents limitations of the state-the-art-method, PMW$^{\text{Pub}}$ [13], via simple pretraining.

2 A Unifying Framework for Private Query Release

In this work, we consider the problem of finding a distribution in some family of distributions $\mathcal{D}$ that achieves low error on all queries. More formally, given a private dataset P and a query set Q, we solve an optimization problem of the form $\min_{D\in\mathcal{D}} \max_{q\in Q} |q(P) - q(D)|$.

In Algorithm 1, we introduce Adaptive Measurements, which serves as a general framework for optimizing this objective. A each round t, the framework uses a private selection mechanism to choose k queries $\widetilde{Q}_t = \{\widetilde{q}_{t,1}, \ldots, \widetilde{q}_{t,k}\}$ with higher error from the set Q. It then obtains noisy measurements for the queries, which we denote by $\widetilde{A}_t = \{\widetilde{a}_{t,1}, \ldots, \widetilde{a}_{t,k}\}$, where $\widetilde{a}_{t,i} = \widetilde{q}_{t,i} + z_{t,i}$ and $z_{t,i}$ is random Laplace or Gaussian noise. Finally, it updates its approximating distribution D_t, subject to a loss function $\mathcal{L}$ that depends on $\widetilde{Q}_{1:t} = \bigcup_{i=1}^{t} \widetilde{Q}_i$ and $\widetilde{A}_{1:t} = \bigcup_{i=1}^{t} \widetilde{A}_i$. We note that $\mathcal{L}$ serves as a surrogate problem to our objective.

Algorithm 1: Adaptive Measurements

Input: Private dataset P with n records, set of linear queries Q, distributional family $\mathcal{D}$, loss functions $\mathcal{L}$, number of iterations T

Initialize distribution $D_0 \in \mathcal{D}$

for $t = 1, \ldots, T$ **do**

Sample: For $i \in [k]$, choose $\widetilde{q}_{t,i}$ using a differentially private selection mechanism.

Measure: For $i \in [k]$, let $\widetilde{a}_{t,i} = \widetilde{q}_{t,i}(P) + z_{t,i}$ where z is Gaussian or Laplace noise

Update: Let $\widetilde{Q}_t = \{\widetilde{q}_{t,1}, \ldots, \widetilde{q}_{t,k}\}$ and $\widetilde{A}_t = \{\widetilde{a}_{t,1}, \ldots, \widetilde{a}_{t,k}\}$. Update distribution D:

$$D_t \leftarrow \arg\min_{D\in\mathcal{D}} \mathcal{L}\left(D, \widetilde{Q}_{1:t}, \widetilde{A}_{1:t}\right)$$

where $\widetilde{Q}_{1:t} = \bigcup_{i=1}^{t} \widetilde{Q}_i$ and $\widetilde{A}_{1:t} = \bigcup_{i=1}^{t} \widetilde{A}_i$.

end

Output $H\left(\{D_t\}_{t=0}^{T}\right)$ where H is some function over all distributions D_t (such as the average)

2.1 Maximum-Entropy Projection Algorithm

In this section, we propose an algorithm (Private Entropy Projection) PEP under the framework Adaptive Measurements. Similar to MWEM, PEP employs the *maximum entropy* principle, which also recovers a synthetic data distribution in the exponential family. However, since PEP adaptively assigns weights to past queries and measurements, it has faster convergence and better accuracy than MWEM. PEP's loss function in Adaptive Measurements can be derived through a constrained maximum entropy optimization problem.

$$\begin{aligned} \text{minimize:} \quad & \sum_{x\in\mathcal{X}} D(x)\log\left(D(x)\right) \\ \text{subject to:} \quad & \forall_{i\in[t]} \quad |\widetilde{a}_i - \widetilde{q}_i(D)| \le \gamma, \quad \sum_{x\in\mathcal{X}} D(x) = 1 \end{aligned} \tag{1}$$

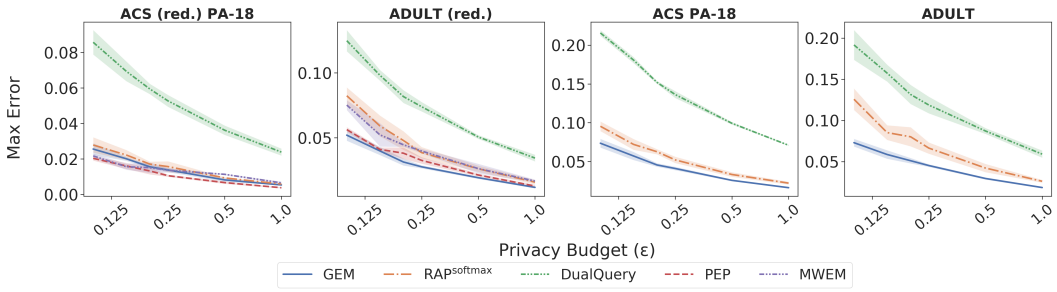


Figure 1: Max error for 3-way marginals evaluated on ADULT and ACS PA-18 using privacy budgets $\varepsilon \in \{0.1, 0.15, 0.2, 0.25, 0.5, 1\}$ and $\delta = \frac{1}{n^2}$. The *x-axis* uses a logarithmic scale. We evaluate using the following workload sizes: ACS (reduced) PA-18: 455; ADULT (reduced): 35; ACS PA-18: 4096; ADULT: 286. Results are averaged over 5 runs, and error bars represent one standard error.

We can ignore the constraint that $\forall_{x \in \mathcal{X}} D(x) \geq 0$, because it will be satisfied automatically. The solution of (1) is an exponentially weighted distribution parameterized by the dual variables $\lambda_1, \ldots, \lambda_t$ corresponding to the t constraints. Therefore, if we solve the dual problem of (1) in terms of the dual variables $\lambda_1, \ldots, \lambda_t$, then the distribution that minimizes (1) is given by $D_t(x) \propto \exp\left(\sum_{i=1}^{t} \lambda_i \widetilde{q}_i(x)\right)$. Given that the set of distributions is parameterized by the variables $\lambda_1, \ldots, \lambda_t$, the constrained optimization is then equivalent to minimizing the following exponential loss function:

$$\mathcal{L}^{\mathsf{PEP}}\left(\lambda, \widetilde{Q}_{1:t}, \widetilde{A}_{1:t}\right) = \log\left(\sum_{x \in \mathcal{X}} \exp\left(\sum_{i=1}^{t} \lambda_i \left(\widetilde{q}_i(x) - \widetilde{a}_i\right)\right)\right) + \gamma\|\lambda\|_1$$

We give an efficient iterative algorithm (Algorithm 2) for solving (1) in Appendix C.

2.2 Overcoming Computational Intractability with Generative Networks

Real-world datasets are often high-dimensional. In such cases, methods like MWEM and PEP suffer from exponential running time, since they explicitly maintain a distribution over the entire data domain $\mathcal{X}$. To alleviate this issue, we introduce GEM, which (like PEP) optimizes over past queries to improve accuracy but (unlike PEP) trains a generator network G_θ to implicitly learn a distribution of the data domain, where G_θ can be any neural network parametrized by weights θ. As a result, our method GEM can compactly represent a distribution for any data domain while enabling fast, gradient-based optimization via auto-differentiation frameworks [17, 1]. Consequently, given some batch size B, we train neural network general G to minimize the loss function

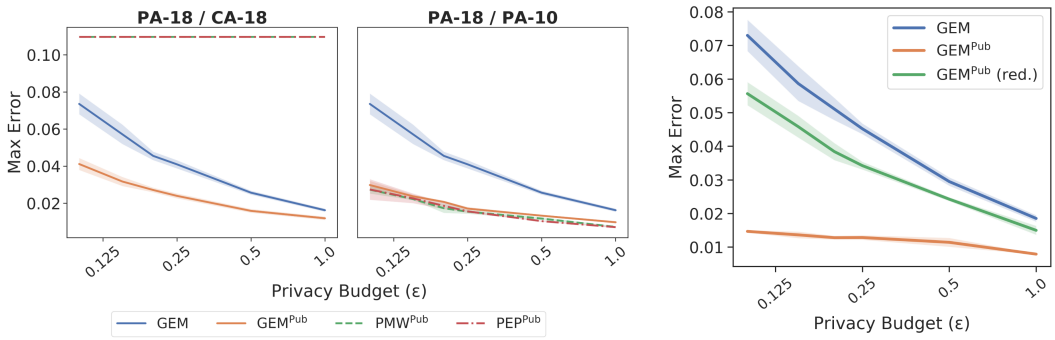
$$\mathcal{L}\left(G, \widetilde{Q}_{1:t}, \widetilde{A}_{1:t}\right) = \sum_{i=1}^{t} \left| \frac{1}{B} \sum_{j=1}^{B} f_{\widetilde{q}_i}\left(G\left(z_j\right)\right) - \widetilde{a}_i \right|. \tag{2}$$

We give additional details for GEM and its optimization problem in the Appendix D

3 Empirical Evaluation

In this section, we empirically evaluate GEM and PEP against baseline methods on the ACS [18] and ADULT [7] datasets in both the standard and *public-data-assisted* settings.

Data. To evaluate our methods, we construct public and private datasets from the ACS and ADULT datasets by following the preprocessing steps outlined in Liu et al. [13]. For the ACS, we use 2018 data for the state of Pennsylvania (PA-18) as the private dataset. For the public dataset, we select 2010 data for Pennsylvania (PA-10) and 2018 data for California (CA-18). In our experiments on the ADULT dataset, private and public datasets are sampled from the complete dataset (using a 90-10 split). In addition, we construct low-dimensional versions of both datasets, which we denote as ACS (reduced) and ADULT (reduced), in order to evaluate PEP and MWEM.



(a) ACS PA-18 (b) ADULT

Figure 2: Max error for 3-way marginals with privacy budgets $\varepsilon \in \{0.1, 0.15, 0.2, 0.25, 0.5, 1\}$ and $\delta = \frac{1}{n^2}$. The *x-axis* uses a logarithmic scale. Results are averaged over 5 runs, and error bars represent one standard error. **(a)** ACS PA-18 (workloads = 4096). We evaluate public-data-assisted algorithms with the following public datasets: *Left:* 2018 California (CA-18); *Right:* 2010 Pennsylvania (PA-10). **(b)** ADULT (workloads = 286). We evaluate GEM using both the complete public data (GEM$^{\text{Pub}}$) and a reduced version that has fewer attributes (GEM$^{\text{Pub}}$ (reduced)) .

Baselines. We compare our algorithms to the strongest performing baselines[2] in both low and high-dimensional settings, presenting results for MWEM, DualQuery, and RAP$^{\text{softmax}}$ in the standard setting and PMW$^{\text{Pub}}$ in the *public-data-assisted* setting.

3.1 Results

Standard setting. In Figure 1, we observe that in low-dimensional settings, PEP and GEM consistently achieve strong performance compared to the baseline methods. While MWEM and PEP are similar in nature, PEP outperforms MWEM on both datasets across all privacy budgets except $\varepsilon \in \{0.1, 0.15\}$ on ACS (reduced), where the two algorithms perform similarly. In addition, both PEP and GEM outperform RAP$^{\text{softmax}}$. Moving on to the more realistic setting in which the data dimension is high, we again observe that GEM outperforms RAP$^{\text{softmax}}$ on both datasets.

***Public-data-assisted* setting.** Next, we evaluate GEM in the public-data-assisted *PAP* [3] setting in which algorithms have access to public data that is free of privacy concerns. We refer to the *PAP* variant of GEM as GEM$^{\text{Pub}}$ (please refer to Appendix E for additional details). We present the three following categories of public data.

Public data with sufficient support. To evaluate our methods when the public dataset for ACS PA-18 has low *best-mixture-error* [13], we consider the public dataset ACS PA-10. We observe in Figure 1 that GEM$^{\text{Pub}}$ performs similarly to PMW$^{\text{Pub}}$, with both outperforming GEM (without public data).

Public data with insufficient support. In Figure 2a, we present CA-18 as an example of this failure case in which the *best-mixture-error* is over 10%, and so for any privacy budget, PMW$^{\text{Pub}}$ cannot achieve max errors lower that this value. However, GEM$^{\text{Pub}}$ is not restricted by *best-mixture-error* and significantly outperforms GEM (without public data) when using either public dataset.

Public data with incomplete data domains. To simulate this setting, we construct a reduced version of the public dataset in which we keep only 7 out of 13 attributes in ADULT. To evaluate GEM$^{\text{Pub}}$, we pretrain the generator G using all 3-way marginals on both the complete and reduced versions of the public dataset and then finetune on the private dataset (we denote these two finetuned networks as GEM$^{\text{Pub}}$ and GEM$^{\text{Pub}}$ (reduced) respectively). We present results in Figure 2b. Given that the public and private datasets are sampled from the same distribution, GEM$^{\text{Pub}}$ unsurprisingly performs extremely well. However, despite only being pretrained on a small fraction of all 3-way marginal queries ($\approx 20k$ out $334k$), GEM$^{\text{Pub}}$ (reduced) is still able to improve upon the performance of GEM and achieve lower max error for all privacy budgets.

[2] RAP performs poorly in our experiments, and so we introduce a variant of RAP called RAP$^{\text{softmax}}$ (see end of Appendix B.2) and compare the two methods in Appendix F.3

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[2] Sergul Aydore, William Brown, Michael Kearns, Krishnaram Kenthapadi, Luca Melis, Aaron Roth, and Ankit Siva. Differentially private query release through adaptive projection. *arXiv preprint arXiv:2103.06641*, 2021.

[3] Raef Bassily, Albert Cheu, Shay Moran, Aleksandar Nikolov, Jonathan R. Ullman, and Zhiwei Steven Wu. Private query release assisted by public data. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 695–703. PMLR, 2020. URL `http://proceedings.mlr.press/v119/bassily20a.html`.

[4] Brett K Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P Bhavnani, James Brian Byrd, and Casey S Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, 12(7): e005122, 2019.

[5] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Proceedings of the 14th Conference on Theory of Cryptography*, TCC '16-B, pages 635–658, Berlin, Heidelberg, 2016. Springer.

[6] Mark Cesar and Ryan Rogers. Unifying privacy loss composition for data analytics. *arXiv preprint arXiv:2004.07223*, 2020.

[7] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[8] Cynthia Dwork and Guy N. Rothblum. Concentrated differential privacy, 2016.

[9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC '06, pages 265–284, Berlin, Heidelberg, 2006. Springer.

[10] Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. Dual query: Practical private query release for high dimensional data. In *International Conference on Machine Learning*, pages 1170–1178. PMLR, 2014.

[11] Moritz Hardt, Katrina Ligett, and Frank Mcsherry. A simple and practical algorithm for differentially private data release. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/208e43f0e45c4c78cafadb83d2888cb6-Paper.pdf`.

[12] Daniel Kifer. Consistency with external knowledge: The topdown algorithm, 2019. `http://www.cse.psu.edu/~duk17/papers/topdown.pdf`.

[13] Terrance Liu, Giuseppe Vietri, Thomas Steinke, Jonathan Ullman, and Zhiwei Steven Wu. Leveraging public data for practical private query release. *arXiv preprint arXiv:2102.08598*, 2021.

[14] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *Proc. VLDB Endow.*, 11(10):1206–1219, June 2018. ISSN 2150-8097. doi: 10.14778/3231751.3231769. URL `https://doi.org/10.14778/3231751.3231769`.

[15] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*, pages 4435–4444. PMLR, 2019.

[16] Marcel Neunhoeffer, Steven Wu, and Cynthia Dwork. Private post-{gan} boosting. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6isfR3JCbi.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[18] S Ruggles et al. Ipums usa: Version 10.0, doi: 10.18128/d010. *V10. 0*, 2020.

[19] Giuseppe Vietri, Grace Tian, Mark Bun, Thomas Steinke, and Steven Wu. New oracle-efficient algorithms for private synthetic data release. In *International Conference on Machine Learning*, pages 9765–9774. PMLR, 2020.

[20] Yasin Yazıcı, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in gan training. *arXiv preprint arXiv:1806.04498*, 2018.

A Preliminaries

Let $\mathcal{X}$ denote a finite d-dimensional data domain (e.g., $\mathcal{X} = \{0,1\}^d$). Lete U be the uniform distribution over the domain $\mathcal{X}$. Throughout we assume a private dataset P that contains the data of n individuals. For any $x \in \mathcal{X}$, we represent $P(x)$ as the normalized frequency of x in dataset P such that $\sum_{x\in\mathcal{X}} P(x) = 1$. One can think of a dataset P either as a multi-set of items from $\mathcal{X}$ or as a distribution over $\mathcal{X}$.

We consider the problem of accurately answering an extensive collection of linear statistical queries (also known as counting queries) about a dataset. Given a finite set of queries Q, our goal is to find a synthetic dataset D such that the maximum error over all queries in Q, defined as $\max_{q\in Q} |q(P) - q(D)|$, is as small as possible. For example, one may query a dataset by asking the following: how many people in a dataset have brown eyes? More formally, a statistical linear query q_ϕ is defined by a predicate function $\phi : \mathcal{X} \to \{0,1\}$, as $q_\phi(D) = \sum_{x\in\mathcal{X}} \phi(x)D(x)$ for any normalized dataset D. Below, we define an important, and general class of linear statistical queries called k-way marginals.

Definition 1 (k-way marginal)**.** *Let the data universe with d categorical attributes be $\mathcal{X} = (\mathcal{X}_1 \times \ldots \times \mathcal{X}_d)$, where each $\mathcal{X}_i$ is the discrete domain of the ith attribute A_i. A k-way marginal query is defined by a subset $S \subseteq [d]$ of k features (i.e., $|S| = k$) plus a target value $y \in \prod_{i\in S} \mathcal{X}_i$ for each feature in S. Then the marginal query $\phi_{S,y}(x)$ is given by:*

$$\phi_{S,y}(x) = \prod_{i\in S} \mathbb{1}\left(x_i = y_i\right)$$

where $x_i \in \mathcal{X}_i$ means the i-th attribute of record $x \in \mathcal{X}$. Each marginal has a total of $\prod_{i=1}^{k} |\mathcal{X}_i|$ queries, and we define a workload as a set of marginal queries.

We consider algorithms that input a dataset P and produce randomized outputs that depend on the data. The output of a randomized mechanism $\mathcal{M} : \mathcal{X}^* \to \mathcal{R}$ is a privacy preserving computation if it satisfies differential privacy (DP) [9]. We say that two datasets are neighboring if they differ in at most the data of one individual.

Definition 2 (Differential privacy [9])**.** *A randomized mechanism $\mathcal{M} : \mathcal{X}^n \to \mathcal{R}$ is (ε, δ)-differentially privacy, if for all neighboring datasets P, P' (i.e., differing on a single person), and all measurable subsets $S \subseteq \mathcal{R}$ we have:*

$$\Pr\left[\mathcal{M}(P) \in S\right] \leq e^{\varepsilon} \Pr\left[\mathcal{M}(P') \in S\right] + \delta$$

Finally, a related notion of privacy is called concentrated differential privacy (zCDP) [8, 5], which enables cleaner composition analyses for privacy.

Definition 3 (Concentrated DP, Dwork and Rothblum [8], Bun and Steinke [5])**.** *A randomized mechanism $\mathcal{M} : \mathcal{X}^n \to \mathcal{R}$ is $\frac{1}{2}\tilde{\varepsilon}^2$-CDP, if for all neighboring datasets P, P' (i.e., differing on a single person), and for all $\alpha \in (1, \infty)$,*

$$\mathrm{R}_\alpha\left(\mathcal{M}(P) \parallel \mathcal{M}(P')\right) \leq \frac{1}{2}\tilde{\varepsilon}^2\alpha$$

where $\mathrm{R}_\alpha\left(\mathcal{M}(P) \parallel \mathcal{M}(P')\right)$ is the Rényi divergence between the distributions $\mathcal{M}(P)$ and $\mathcal{M}(P')$.

B Adaptive Measurements

B.1 Privacy analysis

We present the privacy analysis of the Adaptive Measurements framework while assuming that the exponential and Gaussian mechanism are used for the private *sample* and noisy *measure* steps respectively. More specifically, suppose that we (1) sample k queries using the *exponential mechanism* with the score function:

$$\Pr\left[\widetilde{q}_{t,i} = q\right] \propto \exp\left(\alpha\varepsilon_0 n |q(P) - q(D_{t-1})|\right)$$

and (2) measure the answer to each query by adding Gaussian noise

$$z_{t,i} \sim \mathcal{N}\left(0, \left(\frac{1}{n(1-\alpha)\varepsilon_0}\right)^2\right).$$

Letting $\varepsilon_0 = \sqrt{\frac{2\rho}{T\left(\alpha^2 + (1-\alpha)^2\right)}}$ and $\alpha \in (0,1)$ be a privacy allocation hyperparameter (higher values of α allocate more privacy budget to the exponential mechanism), we present the following theorem:

Theorem 1. *When run with privacy parameter ρ,* Adaptive Measurements *satisfies ρ-zCDP. Moreover for all $\delta > 0$,* Adaptive Measurements *satisfies$(\varepsilon(\delta), \delta)$-differential privacy, where $\varepsilon(\delta) \leq \rho + 2\sqrt{\rho \log(1/\delta)}$.*

We provide a proof sketch for Theorem 1.

Proof sketch. Fix $T \geq 1$ and $\alpha \in (0,1)$. (i) At each iteration $t \in [T]$, Adaptive Measurements runs the exponential mechanism with parameter $2\alpha\varepsilon_0$, which satisfies $\frac{1}{8}\left(2\alpha\varepsilon_0\right)^2 = \frac{1}{2}\left(\alpha\varepsilon_0\right)^2$-*zCDP* [6], and the Gaussian mechanism with parameter $(1-\alpha)\varepsilon_0$, which satisfies $\frac{1}{2}\left[(1-\alpha)\varepsilon_0\right]^2$-*zCDP* [5]. (ii) using the composition theorem for concentrated differential privacy [5], Adaptive Measurements satisfies $\frac{T}{2}\left[\alpha^2 + (1-\alpha)^2\right]\varepsilon_0^2$-*zCDP* after T iterations. (iii) Setting $\varepsilon_0 = \sqrt{\frac{2\rho}{T\left(\alpha^2 + (1-\alpha)^2\right)}}$, we conclude that Adaptive Measurements satisfies ρ-*zCDP*, which in turn implies $\left(\rho + 2\sqrt{\rho\log(1/\delta)}, \delta\right)$-*differential privacy* for all $\delta > 0$ [5].

B.2 Choices of loss functions and distributional families

We note that in addition to the loss function $\mathcal{L}$, a key component that differentiates algorithms under this framework is the distributional family $\mathcal{D}$ that the output of each algorithm belongs to. Below, we describe in more detail how existing algorithms fit into our general framework under different choices of $\mathcal{L}$ and $\mathcal{D}$.

MWEM **from Hardt et al. [11]** The traditional MWEM algorithm samples one query each round, thus after t round the set of queries/measurements is $\widetilde{Q}_t = \{\widetilde{q}_1, \ldots, \widetilde{q}_t\}, \widetilde{A}_t = \{\widetilde{a}_1, \ldots, \widetilde{a}_t\}$. Then, on round t, MWEM solves an entropy regularized problem. Let the $t-1$ previous solutions be $D_1, \ldots D_{t-1}$, then MWEM finds D_t by minimizing the following loss:

$$\mathcal{L}^{\text{MWEM}}(D, \widetilde{Q}_{1:t}, \widetilde{A}_{1:t}) = \sum_{i=1}^{t}\sum_{x\in\mathcal{X}} D(x)\widetilde{q}_i(x)\left(\widetilde{a}_i - \widetilde{q}_i(D_{i-1})\right) + \sum_{x\in\mathcal{X}} D(x)\log D(x)$$

We can show that if $D_t = \arg\min_{D\in\Delta(\mathcal{X})} \mathcal{L}^{\text{mwem}}(D, \widetilde{Q}_t, \widetilde{A}_t)$ then D_t evaluates to $D_t(x) \propto \exp\left(-\sum_{i=1}^{t}\widetilde{q}_i(x)(\widetilde{a}_i - \widetilde{q}_i(D_{i-1}))\right)$ which is the exactly the distribution computed by MWEM. See B.3 for derivation. We note that MWEM explicitly maintains (and outputs) a distribution $D \in \mathcal{D}$ where $\mathcal{D}$ includes all distributions over the data domain $\mathcal{X}$, making it computationally intractable for high-dimension settings.

DualQuery **from Gaboardi et al. [10]** DualQuery is a special case of the Adaptive Measurements framework in which the measurement step is skipped (abusing notation, we say $\alpha = 1$). Over all iterations of the algorithm, DualQuery keeps track of a

probability distribution over the set of queries $\mathcal{Q}$ via multiplicative weights, which we denote here by $\mathcal{Q}_t \in \Delta(Q)$. On round t, DualQuery samples s queries ($\widetilde{Q}_t = \{\widetilde{q}_{t,1}, \ldots \widetilde{q}_{t,s}\}$) from $\mathcal{Q}_t$ and outputs D_t that minimizes the the following loss function:

$$\mathcal{L}^{\mathsf{DualQuery}}(D, \widetilde{Q}_t) = \sum_{i=1}^{s} \widetilde{q}_{t,i}(D)$$

The optimization problem for $\mathcal{L}^{\mathsf{DualQuery}}(D, \widetilde{Q}_t)$ is NP-hard. However, the algorithm encodes the problem as a mix-integer-program (MIP) and takes advantage of available fast solvers. The final output of DualQuery is the average $\frac{1}{T}\sum_{t=1}^{T} D_t$, which we note implicitly describes some empirical distribution over $\mathcal{X}$.

FEM from Vietri et al. [19] The algorithm FEM follows a follow the perturbed leader strategy. As with MWEM, the algorithm FEM samples one query each round using the exponential mechanism, so that the set of queries in round t is $\widetilde{Q}_t = \{\widetilde{q}_1, \ldots, \widetilde{q}_t\}$. Then on round t, FEM chooses the next distribution by solving:

$$\mathcal{L}^{\mathsf{FEM}}(D, \widetilde{Q}_{1:t}) = \sum_{i=1}^{t} \widetilde{q}_t(D) + \mathbb{E}_{x\sim D, \eta\sim \text{Exp}(\sigma)^d}\left(\langle x, \eta\rangle\right)$$

Similar to DualQuery, the optimization problem for $\mathcal{L}^{\mathsf{FEM}}$ also involves solving an NP-hard problem. Additionally, the function $\mathcal{L}^{\mathsf{FEM}}$ does not have a close form due to the expectation term, therefore FEM follows a sampling strategy to approximate the optimal solution. On each round FEM generates s samples, with each sample is obtained as: Sample a noise vector $\eta \sim \text{Exp}(\sigma)^d$ from the exponential distribution and use a MIP to solve $x_{t,i} \leftarrow \arg\min_{x\in\mathcal{X}} \sum_{i=1}^{t} \widetilde{q}_t(D) + \langle x, \eta\rangle$ for all $i \in [s]$. Finally, the output on round t is the empirical distribution derived from the s samples: $D_t = \{x_{t,1}, \ldots, x_{t,s}\}$. The final output is the average $\frac{1}{T}\sum_{t=1}^{T} D_t$.

RAP$^{\text{softmax}}$ adapted from Aydore et al. [2] We note that RAP follows the basic structure of Adaptive Measurements, where at iteration t, RAP solves the following optimization problem:

$$\mathcal{L}^{\mathsf{RAP}}(D, \widetilde{Q}_{1:t}, \widetilde{A}_{1:t}) = \sum_{i,j} \left(\widetilde{q}_{i,j}(D) - \widetilde{a}_{i,j}\right)^2$$

However, rather than outputting a dataset that can be expressed as some distribution over $\mathcal{X}$, RAP projects the noisy measurements onto a continuous relaxation of the binarized feature space of $\mathcal{X}$, outputting $D \in [-1, 1]^{n'\times d}$ (where n' is an additional parameter). Therefore to adapt RAP to Adaptive Measurements, we propose a new baseline algorithm that applies the softmax function instead of clipping each dimension of D to be between -1 and 1. For more details, refer to the Appendix D, where describe how softmax is applied in GEM in the same way. With this slight modification, this algorithm, which we denote as RAP$^{\text{softmax}}$, fits nicely into the Adaptive Measurements framework in which we output a synthetic dataset drawn from some probabilistic family of distributions $\mathcal{D} = \left\{\sigma(M) | M \in \mathbb{R}^{n'\times d}\right\}$.

B.3 MWEM loss function derivation

Fix round t, let $\eta_t = \widetilde{a}_t - \widetilde{q}_t(D_{i-1})$ be the learning rate during round t and D_{t-1} be the distribution from the previous round. Then MWEM's loss function is:

$$\mathcal{L}^{\text{mwem}}(D, \widetilde{Q}_t, \widetilde{A}_t) = \eta_t \sum_{x\in\mathcal{X}} D(x)\widetilde{q}_i(x) + \sum_{x\in\mathcal{X}} D(x) \log\left(\frac{D(x)}{D_{t-1}(x)}\right) \tag{3}$$

The optimization problem becomes $D_t = \arg\min_{D\in\Delta(\mathcal{X})} \mathcal{L}^{\text{mwem}}(D, \widetilde{Q}_t, \widetilde{A}_t)$. Since the solution is a distribution, then this problem is a constrained optimization problem. We can express the constrain that the solution D is a distribution by $\sum_{x\in\mathcal{X}} D(x) = 1$. To show that (3) is the MWEM's true loss

function, we can write down the Lagrangian which is given by:

$$\mathcal{L} = \eta_t \sum_{x \in \mathcal{X}} D(x) \widetilde{q}_i(x) + \sum_{x \in \mathcal{X}} D(x) \log \left(\frac{D(x)}{D_{t-1}(x)} \right) + \lambda \left(\sum_{x \in \mathcal{X}} D(x) - 1 \right)$$

Taking partial derivative with respect to $D(x)$:

$$\frac{\partial \mathcal{L}}{\partial D(x)} = \eta_t \widetilde{q}_i(x) + 1 + \log \frac{D(x)}{D_{t-1}(x)} + \lambda$$

Setting $\frac{\partial \mathcal{L}}{\partial D(x)} = 0$ and solving for $D(x)$:

$$D(x) = D_{t-1}(x) \exp\left(-1 - \lambda - \eta_t \widetilde{q}_i(x)\right)$$

Finally, the value of λ is set such that D is a probability distribution:

$$D(x) = \frac{D_{t-1}(x) \exp\left(-\eta_t \widetilde{q}_t(x)\right)}{\sum_{x \in \mathcal{X}} D_{t-1}(x) \exp\left(-\eta_t \widetilde{q}_t(x)\right)}$$

That concludes the derivation of MWEM's loss function.

C PEP

Algorithm 2: Exponential Weights Projection

Input: Error tolerance γ, linear queries $Q = \{\widetilde{q}_1, \ldots, \widetilde{q}_T\}$, and noisy measurements $\{\widetilde{a}_1, \ldots, \widetilde{a}_T\}$.
Objective: Minimize $\text{RE}(D \parallel U)$ such that $\forall_{i\in[T]} \quad |\widetilde{q}_i(D) - \widetilde{a}_i| \leq \gamma$.
Initialize D_0 to be the uniform distribution over $\mathcal{X}$, and $t \leftarrow 0$.
while $\max_{i\in[T]} |\widehat{a}_i - \widetilde{q}_i(D_t)| > \gamma$ **do**
  Choose: $i \in [T]$ with $i \leftarrow \arg\max_{j\in[T]} |\widetilde{a}_j - \widetilde{q}_j(D_t)|$.
  Update: For all $x \in \mathcal{X}$, set $D_{t+1}(x) \leftarrow D_t(x)e^{-\alpha_t \widetilde{q}_i(x)}$, where $-\alpha_t = \ln\left(\frac{(\widetilde{a}_i)(1-\widetilde{q}_i(D_t))}{(1-\widetilde{a}_i)\widetilde{q}_i(D_t)}\right)$.
  $t \leftarrow t + 1$
end
Output: D_T

We give the exact details of PEP in Algorithm 2

C.1 PEP derivations

In this section we derive the loss function that algorithm PEP optimizes over on round t. Fix round t, given a subset of queries $\widetilde{Q}_t \subset Q$ that were selected using a private mechanism and noisy measurements $\widetilde{A}_t$. Then algorithm PEP finds a feasible solution to problem:

$$\begin{aligned} \text{minimize:} \quad & \text{RE}(D \parallel U) \\ \text{subject to:} \quad & \forall_{i\in[t]} \quad |\widetilde{a}_i - \widetilde{q}_i(D)| \leq \gamma, \quad \sum_{x\in\mathcal{X}} D(x) = 1 \end{aligned} \tag{4}$$

Suppose that we run PEP with zCDP privacy parameter ρ and T rounds, then by composition each round mush satisfy $\frac{\rho}{T}$-zCDP. Therefore the noisy measurements are given by For every query $q \in \widetilde{Q}_t$, by the properties of the Gaussian mechanism we have that $|\widetilde{a}_i - \widetilde{q}_i(P)| \leq \frac{|\widetilde{Q}_t|\log(|Q|/\beta)}{\varepsilon}$ for all $i \in [t]$ with probability at least $1 - \beta$. Let $\gamma = \frac{\log(|Q|/\beta)}{\varepsilon}$, then our algorithm solves the following constrained optimization problem using iterative projection:

The Lagrangian of 4 is :

$$\mathcal{L} = \text{RE}(D \parallel U) + \sum_{i=1}^{t} \lambda_i^+ \left(\widetilde{a}_i - \widetilde{q}_i(D) - \gamma\right) + \sum_{i=1}^{t} \lambda_i^- \left(\widetilde{q}_i(D) - \widetilde{a}_i - \gamma\right) + \mu\left(\sum_x D(x) - 1\right)$$

Let $\lambda \in \mathbb{R}^t$ be a vector with, $\lambda_i = \lambda_i^- - \lambda_i^+$, then

$$\mathcal{L} = \text{RE}(D \parallel U) + \sum_{i=1}^{t} \lambda_i \widetilde{q}_i(D) - \sum_{i=1}^{t} \lambda_i \widetilde{a}_i - \gamma \sum_{i=1}^{t} \left(\lambda_i^+ + \lambda_i^-\right) + \mu\left(\sum_x D(x) - 1\right) \tag{5}$$

where $\|\lambda\|_1 = \sum_{i=1}^{t} \left(\lambda_i^+ + \lambda_i^-\right)$.

Taking the derivative with respect to $D(x)$ and setting to zero we get:

$$0 = \frac{\partial \mathcal{L}}{\partial D(x)} = \log\left(\frac{D(x)}{U(x)}\right) + 1 + \sum_{i=1}^{t} \lambda_i \widetilde{q}_i(x) + \mu$$

Then for any $x \in \mathcal{X}$ we have

$$D(x) = U(x) \exp\left(-\sum_{i=1}^{t} \lambda_i \widetilde{q}_i(x) - \mu - 1\right)$$

The slack variable μ must be selected to satisfy the constrain that $\sum_{x\in\mathcal{X}} D(x) = 1$, therefore we get:

$$D(x) = \frac{U(x)}{Z} \exp\left(-\sum_{i=1}^{t} \lambda_i \widetilde{q}_i(x)\right)$$

where $Z = \sum_{x \in \mathcal{X}} U(x) \exp\left(-\sum_{i=1}^{t} \lambda_i \widetilde{q}_i(x)\right)$. Plugging into (5) we get

$$\begin{aligned}
\mathcal{L} &= \sum_{x \in \mathcal{X}} D(x) \log\left(\frac{D(x)}{U(x)}\right) + \sum_{i=1}^{t} \lambda_i \widetilde{q}_i(D) - \sum_{i=1}^{t} \lambda_i \widetilde{a}_i - \gamma \|\lambda\|_1 \\
&= \sum_{x \in \mathcal{X}} D(x) \left(-\sum_{i=1}^{t} \lambda_i \widetilde{q}_i(x)\right) - \log(Z) + \sum_{i=1}^{t} \lambda_i \widetilde{q}_i(D) - \sum_{i=1}^{t} \lambda_i \widetilde{a}_i - \gamma \|\lambda\|_1 \\
&= -\sum_{i=1}^{t} \lambda_i \widetilde{q}_i(D) - \log(Z) + \sum_{i=1}^{t} \lambda_i \widetilde{q}_i(D) - \sum_{i=1}^{t} \lambda_i \widetilde{a}_i - \gamma \|\lambda\|_1 \\
&= -\log(Z) + \sum_{i=1}^{t} \lambda_i \widetilde{a}_i - \gamma \|\lambda\|_1 \\
&= -\log\left(\frac{Z}{\exp\left(\sum_{i=1}^{t} \lambda_i \widetilde{a}_i\right)}\right) - \gamma \|\lambda\|_1
\end{aligned}$$

Substituting the Z we get:

$$\begin{aligned}
\mathcal{L} &= -\log\left(\frac{\sum_{x \in \mathcal{X}} \exp\left(\sum_{i=1}^{t} \lambda_i \widetilde{q}_i(x)\right)}{\exp\left(\sum_{i=1}^{t} \lambda_i \widetilde{a}_i\right)}\right) - \gamma \|\lambda\|_1 \\
&= -\log\left(\sum_{x \in \mathcal{X}} \exp\left(\sum_{i=1}^{t} \lambda_i \left(\widetilde{q}_i(x) - \widetilde{a}_i + \gamma\right)\right)\right) - \gamma \|\lambda\|_1
\end{aligned}$$

Finally, we have that the dual problem of (4)is finding a vector $\lambda = (\lambda_1, \ldots, \lambda_t)$ that maximizes $\mathcal{L}$. We can write the dual problem as a minimization problem:

$$\mathcal{L}(\lambda) = \min_{\lambda} \log\left(\sum_{x \in \mathcal{X}} \exp\left(\sum_{i=1}^{t} \lambda_i \left(\widetilde{q}_i(x) - \widetilde{a}_i\right)\right)\right) + \gamma \|\lambda\|_1$$

D GEM

Algorithm 3: GEM

Input: Private dataset P, set of differentiable queries Q
Parameters: privacy parameter ρ, number of iterations T, privacy weighting parameter α, batch size B, stopping threshold γ
Initialize generator network G_0
Let $\varepsilon_0 = \sqrt{\frac{2\rho}{T\left(\alpha^2+(1-\alpha)^2\right)}}$
for $t = 1 \dots T$ **do**
  Sample: Sample $\mathbf{z} = \langle z_1 \dots z_B \rangle \sim \mathcal{N}(0, I_B)$
  Choose $\widetilde{q}_t$ using the *exponential mechanism* with score

$$\Pr\left[q_t = q\right] \propto \exp\left(\frac{\alpha\varepsilon_0 n}{2}|q(P) - q(G_{t-1}\left(\mathbf{z}\right))|\right)$$

  Measure: Let $\widetilde{a}_t = \widetilde{q}_t(P) + \mathcal{N}\left(0, \left(\frac{1}{n(1-\alpha)\varepsilon_0}\right)^2\right)$
  Update: $G_t = \mathsf{GEM\text{-}UPDATE}\left(G_{t-1}, Q_t, \widetilde{\mathbf{a}}_\mathbf{t}, \gamma\right)$ where $Q_t = \langle \widetilde{q}_1, \dots, \widetilde{q}_t \rangle$ and $\widetilde{\mathbf{a}}_\mathbf{t} = \langle \widetilde{a}_1, \dots, \widetilde{a}_t \rangle$
end
Let $\theta_{out} = \mathsf{EMA}\left(\{\theta_j\}_{j=\frac{T}{2}}^{T}\right)$ where θ_j parameterizes G_j
Let G_{out} be the generator parameterized by θ_{out}
Output $G_{out}\left(\mathbf{z}\right)$

Algorithm 4: GEM-UPDATE

Input: Generator G parameterized by θ, queries Q, noisy measurements $\widetilde{\mathbf{a}}$, stopping threshold γ
Parameters: max iterations T_{max}, batch size B
Sample $\mathbf{z} = \langle z_1 \dots z_B \rangle \sim \mathcal{N}(0, I_B)$
Let $\mathbf{c} = \widetilde{\mathbf{a}} - \frac{1}{B}\sum_{j=1}^{B} f_Q\left(G\left(z_j\right)\right)$ be errors over queries Q
Let $i = 0$
while $i < T_{max}$ *and* $\|\mathbf{c}\|_\infty \geq \gamma$ **do**
  Let $J = \{j \mid |c_j| \geq \gamma\}$
  Update G to minimize the loss function with the stochastic gradient $\nabla_\theta \frac{1}{|J|}\sum_{j\in J}|c_{ij}|$
  Sample $\mathbf{z} = \langle z_1 \dots z_B \rangle \sim \mathcal{N}(0, I_B)$
  Let $\mathbf{c} = \widetilde{\mathbf{a}} - \frac{1}{B}\sum_{j=1}^{B} f_Q\left(G\left(z_j\right)\right)$
  Let $i = i + 1$
end
Output: G

Concretely, G_θ takes random Gaussian noise vectors z as input and outputs a representation $G_\theta(z)$ of a product distribution over the data domain. Specifically, this product distribution representation takes the form of a d'-dimensional probability vector $G_\theta(z) \in [0,1]^{d'}$, where d' is the dimension of the data in one-hot encoding and each coordinate $G_\theta(z)_j$ corresponds to the marginal probability of a categorical variable taking on a specific value. To obtain this probability vector, we choose softmax as the activation function for the output layer in G_θ. Therefore, for any fixed weights θ, G_θ defines a distribution over $\mathcal{X}$ through the generative process that draws a random $z \sim \mathcal{N}(0, \sigma^2 I)$ and then outputs random x drawn from the product distribution $G_\theta(z)$. We will denote this distribution as P_θ.

To define the loss function for GEM, recall first that a query $q : \mathcal{X} \to \{0,1\}$ is function defined over the data domain $\mathcal{X}$ that evaluates over a single row $x \in \mathcal{X}$. In order to use gradient-based methods to optimize G_θ, we need to obtain a differentiable variant of q. We observe that one can extend any statistical query q to be a function that maps a distribution P_θ over $\mathcal{X}$ to a value in $[0,1]$:

$$q(P_\theta) = \mathbb{E}_{x\sim P_\theta}\left[q(x)\right] = \sum_{x\in\mathcal{X}} P_\theta(x)q(x) \tag{6}$$

Note that any statistical query q is then differentiable w.r.t. θ:

$$\nabla_\theta q(P_\theta) = \sum_{x \in \mathcal{X}} \nabla P_\theta(x) q(x) = \sum_{x \in \mathcal{X}} \mathbb{E}_z \left[\nabla \prod_{j=1}^{d'} (G_\theta(z)_j)^{x_j} \right] q(x)$$

and we can compute stochastic gradients of q w.r.t. θ with random noise samples z. This also allows us to derive a differentiable loss function in the Adaptive Measurements framework. In each round t, given a set of selected queries $\widetilde{Q}_{1:t}$ and their noisy measurements $\widetilde{A}_{1:t}$, GEM minimizes the following ℓ_1-loss:

$$\mathcal{L}^{\mathsf{GEM}}\left(\theta, \widetilde{Q}_{1:t}, \widetilde{A}_{1:t}\right) = \sum_{i=1}^{t} \left| \widetilde{q}_i(P_\theta) - \widetilde{a}_i \right|. \tag{7}$$

In general, we can optimize $\mathcal{L}^{\mathsf{GEM}}$ by running stochastic (sub)-gradient descent, but remark that gradient computation can be expensive. To obtain a low-variance gradient estimate, one needs to estimate the gradient $\nabla_\theta P_\theta(x)$ for a large number of x.

For many query classes, however, there exists some closed-form, differentiable function surrogate to (7) that evaluates $q(G_\theta(z))$ directly without operating over all $x \in \mathcal{X}$. Concretely, we say that for certain query classes, there exists some representation $f_q : \Delta(\mathcal{X}) \to [0, 1]$ for q that operates in the probability space of $\mathcal{X}$ and is also differentiable.

In this work, we implement GEM to answer k-way marginal queries, which has been one of the most important query classes for the query release literature [11, 19, 10, 13] and provides a differentiable form when extended to be a function over distributions. In particular, we show that k-way marginals can be rewritten as product queries (which are differentiable).

Definition 4 (Product query)**.** *Let $p \in \mathbb{R}^{d'}$ be a representation of a dataset (in the one-hot encoded space), and let $S \subseteq [d']$ be some subset of dimensions of p. Then we define a product query f_S as*

$$f_S(p) = \prod_{j \in S} p_j \tag{8}$$

A k-way marginal query ϕ can then be rewritten as (8), where $p = G_\theta(z)$ and S is the subset of dimensions corresponding to the attributes A and target values y that are specified by ϕ (Definition 1). Thus, we can write any marginal query as $\prod_{j \in S} G_\theta(z)_j$, which is differentiable w.r.t. G_θ (and therefore differentiable w.r.t weights θ by chain rule). Gradient-based optimization techniques can then be used to solve (7). We show the exact details of GEM Algorithms 3 and 4. Note that given a vector of queries $Q_t = \langle q_1, \ldots, q_t \rangle$, we define $f_{Q_t}(\cdot) = \langle f_{q_1}(\cdot), \ldots, f_{q_t}(\cdot) \rangle$.

D.1 Loss function (for k-way marginals)

For any $z \in \mathbb{R}$, $G(z)$ outputs a distribution over each attribute, which we can use to calculate the answer to a query via f_q. In GEM however, we instead sample a noise vector $\mathbf{z} = \langle z_1 \ldots z_B \rangle$ and calculate the answer to some query q as $\frac{1}{B} \sum_{j=1}^{B} f_q\left(G\left(z_j\right)\right)$. One way of interpreting the batch size B is to consider each $G\left(z_j\right)$ as a unique distribution. In this sense, GEM models B sub-populations that together comprise the overall population of the synthetic dataset. Using this notation, GEM outputs then a generator $G \in \mathcal{D}$ by optimizing ℓ_1-loss at each step t of the Adaptive Measurements framework:

$$\mathcal{L}\left(G, \widetilde{Q}_{1:t}, \widetilde{A}_{1:t}\right) = \sum_{i=1}^{t} \left| \frac{1}{B} \sum_{j=1}^{B} f_{\widetilde{q}_i}\left(G\left(z_j\right)\right) - \widetilde{a}_i \right| \tag{9}$$

D.2 EMA output

We observe empirically that the performance of the last generator G_T is often unstable. One possible solution explored previously in the context of privately trained GANs is to output a mixture of samples from a set of generators [4, 16]. In our algorithm GEM, we instead draw inspiration from Yazıcı et al. [20] and output a single generator G_{out} whose weights θ_{out} are an exponential moving

average (EMA) of weights θ_t obtained from the latter half of training. More concretely, we define $\theta_{out} = \mathsf{EMA}\left(\{\theta_j\}_{j=\frac{T}{2}}^{T}\right)$, where the update rule for EMA is given by $\theta_k^{EMA} = \beta\theta_{k-1}^{EMA} + (1-\beta)\theta_k$ for some parameter β.

D.3 Stopping threshold γ

To reduce runtime and prevent GEM from overfitting to the set of queries sampled, we run GEM-UPDATE with some early stopping threshold set to an error tolerance γ. Empirically, we find that setting γ to be half of the max error at each time step t. Because sampling the max query using the exponential mechanism provides a noisy approximation of the true max error, we find that using an exponential moving average (with $\beta = 0.5$) of the sampled max errors is a more stable approximation of the true max error. More succinctly, we set $\gamma = \mathsf{EMA}(\{c_i\}_{i=0}^{t})$ where c_i is max error at iteration i.

E Extending to the *public-data-assisted* setting

Incorporating prior information from public data has shown to be a promising avenue for private query release [3, 13]. Therefore we extend PEP and GEM to the problem of *public-data-assisted private (PAP)* query release [3] in which differentially private algorithms have access to public data.

PEP$^{\text{Pub}}$. Like in Liu et al. [13], we extend PEP by making two changes: (1) we maintain a distribution over the public data domain and (2) we initialize the approximating distribution to that of the public dataset. Therefore like PMW$^{\text{Pub}}$, PEP$^{\text{Pub}}$ also restricts $\mathcal{D}$ to distributions over the public data domain and initializes D_0 to be the public data distribution.

GEM. We adapt GEM to utilize public data by initializing D_0 to a distribution over the public dataset. However, because in GEM, we implicitly model any given distribution using a generator G, we must first train without privacy (i.e., without using the exponential and Gaussian mechanisms) a generator G_{pub}, to minimize the ℓ_1-error over some set of queries $\hat{Q}$. Note that in most cases, we can simply let $\hat{Q} = Q$ where Q is the collection of statistical queries we wish to answer privately. GEM$^{\text{Pub}}$ then initializes G_0 to G_{pub} and proceeds with the rest of the GEM algorithm.

E.1 Overcoming limitations of PMW$^{\text{Pub}}$.

We describe the limitations of PMW$^{\text{Pub}}$ by providing two example categories of public data that it fails to use effectively. We then describe how GEM$^{\text{Pub}}$ overcomes such limitations in both scenarios.

Public data with insufficient support. We first discuss the case in which the public dataset has an insufficient support, which in this context means the support has high *best-mixture-error* [13]. Given some support $S \subseteq \mathcal{X}$, the *best-mixture-error* can be defined as

$$\min_{\mu \in \Delta(S)} \max_{q \in Q} \left| q(D) - \sum_{x \in S} \mu_x q(x) \right|$$

where $\mu \in \Delta(S)$ is a distribution over the set S with $\mu(x) \geq 0$ for all $x \in S$ and $\sum_{x \in S} \mu(x) = 1$.

In other words, the *best-mixture-error* approximates the lowest possible max error that can be achieved by reweighting some support, which in this case means PMW$^{\text{Pub}}$ cannot achieve max errors lower that this value. While Liu et al. [13] offer a solution for filtering out poor public datasets ahead of time using a small portion of the privacy budget, PMW$^{\text{Pub}}$ cannot be run effectively if no other suitable public datasets exist. GEM$^{\text{Pub}}$ however avoids this issue altogether because unlike MWEM (and therefore PMW$^{\text{Pub}}$), which cannot be run without restricting the size of $\mathcal{D}$, GEM$^{\text{Pub}}$ can utilize public data without restricting the distributional family it can represent (since both GEM and GEM$^{\text{Pub}}$ compactly parametrize any distribution using a neural network).

Public data with incomplete data domains. Next we consider the case in which the public dataset only has data for a subset of the attributes found in the private dataset. We note that as presented in Liu et al. [13], PMW$^{\text{Pub}}$ cannot handle this scenario. One possible solution is to augment the public data distribution by assuming a uniform distribution over all remaining attributes missing in the public dataset. However, while this option may work in cases where only a few attributes are missing, the missing support grows exponentially in the dimension of the missing attributes. In contrast, GEM$^{\text{Pub}}$ can still make use of such public data. In particular, we can pretrain a generator G on queries over just the attributes found in the public dataset. Again, GEM$^{\text{Pub}}$ avoids the computational intractability of PMW$^{\text{Pub}}$ in this setting since it parametrizes its output distribution with G.

F Additional empirical evaluation

F.1 Experimental details

To present a fair comparison, we implement all algorithms using the privacy mechanisms and *zCDP* composition described in Section B.1. To implement GEM for k-way marginals, we select a simple multilayer perceptron for G_θ. Our implementations of MWEM and PMW$^{\text{Pub}}$ output the last iterate D_t instead of the average and apply the multiplicative weights update rule using past queries according to the pseudocode described in Liu et al. [13]. We report the best performing 5-run average across hyperparameter choices for each algorithm.

We present hyperparameters used for methods across all experiments in Tables 1, 2, and 3. Our implementaions of MWEM, DualQuery, and PMW$^{\text{Pub}}$ are adapted from `https://github.com/terranceliu/pwm-pub`. We implement RAP and RAP$^{\text{softmax}}$ ourselves using PyTorch. All experiments are run using a personal desktop computer with an Intel® Core™ i5-4690K processor and NVIDIA GeForce GTX 1080 Ti graphics card.

We obtain the ADULT and ACS datasets by following the instructions outlined in `https://github.com/terranceliu/pwm-pub`. Our version of ADULT used to train GEM$^{\text{Pub}}$ (reduced) uses the following attributes: sex, race, relationship, marital-status, occupation, education-num, age.

Table 1: PEP hyperparameters

Dataset	Parameter	Values
All	T_{max}	25
ACS (red.)	T	20, 30, 40, 50, 75 100, 125, 150, 175, 200
ADULT (red.)	T	20, 30, 40, 50, 75 100, 125, 150, 175, 200

Table 2: GEM hyperparameters

Dataset	Parameter	Values
All	hidden layer sizes	$(512, 1024, 1024)$
	learning rate	0.0001
	m	1000
	α	0.67
	T_{max}	100
ACS	T	100, 150, 200, 250, 300, 400, 500, 750, 1000
ACS (red.)	T	50, 75, 100, 125, 150, 200, 250, 300
ADULT, ADULT (red.), ADULT (orig), LOANS	T	30, 40, 50, 60, 70, 80, 90, 100, 125, 150, 175, 200

F.2 Main experiments with additional metrics

In Figures 3, 4, and 5, we present the same results for the same experiments described in Figures 1 and 2, adding plots for mean error and root mean squared error (RMSE). For our experiments on ACS PA-18 with public data, we also add results using 2018 data for Ohio (ACS OH-18), which we note also low *best-mixture-error*. We note that generally, the relative performance between the methods for these other two metrics is the same as for max error.

In addition, in Figure 4, we present results for PEP$^{\text{Pub}}$, a version of PEP similar to PMW$^{\text{Pub}}$ that is adapted to leverage public data (and consequently can be applied to high dimensional settings). We

Table 3: Baseline hyperparameters

Method	Parameter	Values
RAP	learning rate	0.001
	n'	1000
	K	5, 10, 25, 50, 100
	T	2, 5, 10, 25, 50, 75, 100
RAP$^{\text{softmax}}$	learning rate	0.1
	n'	1000
	K	5, 10, 25, 50, 100
	T	2, 5, 10, 25, 50, 75, 100
MWEM	T	100, 150, 200, 250, 300 400, 500, 750, 1000
MWEM (/w past queries)	T	50, 75, 100, 150, 200, 250, 300
DualQuery	η	2, 3, 4, 5
	samples	25 50, 100, 250, 500

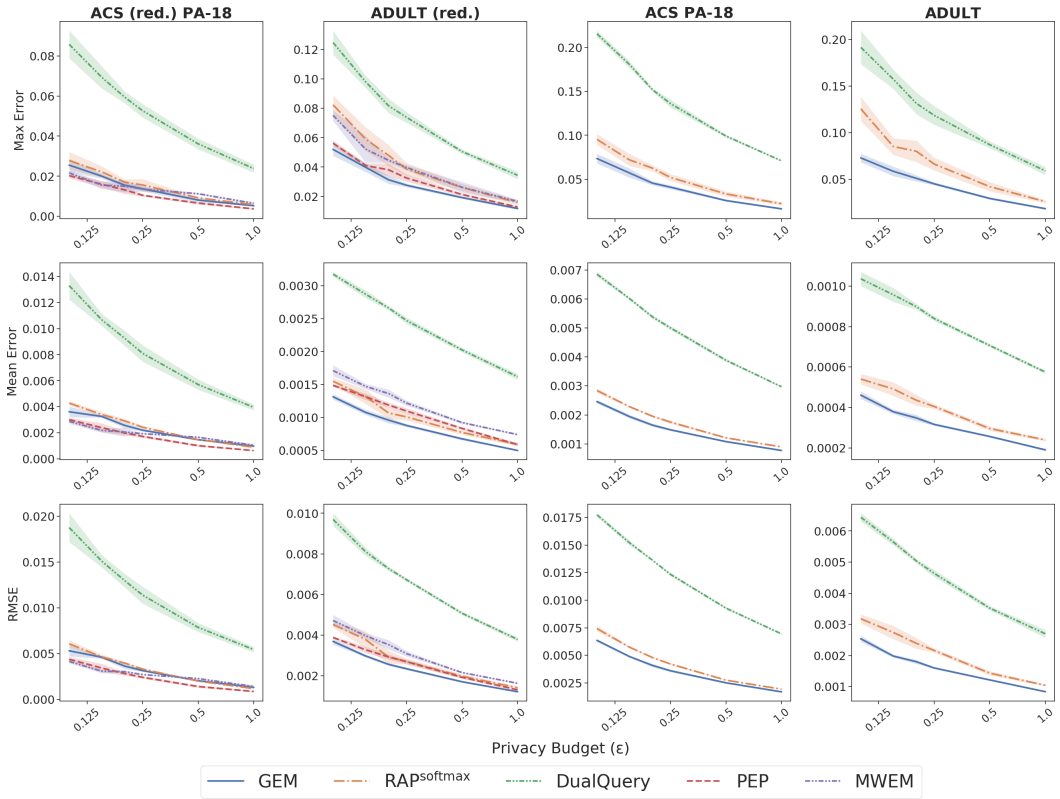


Figure 3: Max, mean, and root mean squared errors for 3-way marginals evaluated on ADULT and ACS PA-18 using privacy budgets $\varepsilon \in \{0.1, 0.15, 0.2, 0.25, 0.5, 1\}$ and $\delta = \frac{1}{n^2}$. The *x-axis* uses a logarithmic scale. We evaluate using the following workload sizes: ACS (reduced) PA-18: 455; ADULT (reduced): 35; ACS PA-18: 4096; ADULT: 286. Results are averaged over 5 runs, and error bars represent one standard error.

note that PEP$^{\text{Pub}}$ performs similarly to PMW$^{\text{Pub}}$, making it unable to perform well when using ACS CA-18 as a public dataset (for experiments on ACS PA-18). Similarly, it cannot be feasibly run for the ADULT dataset when the public dataset is missing a significant number of attributes.

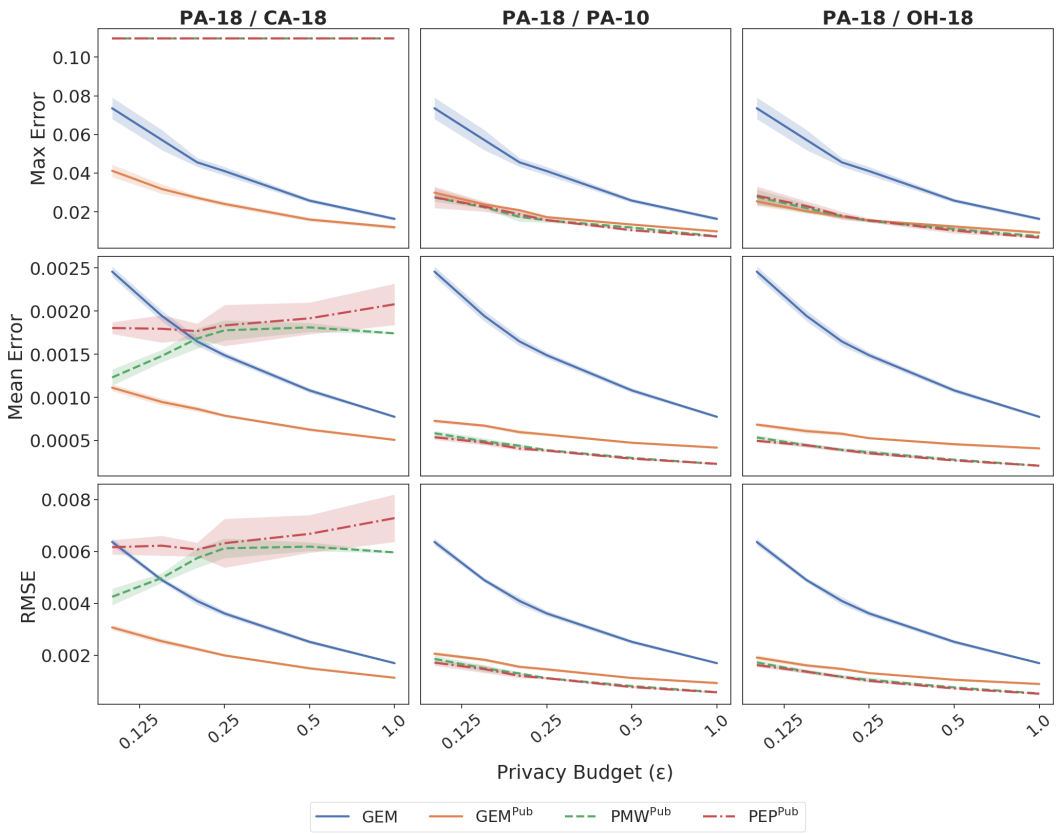


Figure 4: Max, mean, and mean squared error for 3-way marginals on ACS PA-18 (workloads = 4096) with privacy budgets $\varepsilon \in \{0.1, 0.15, 0.2, 0.25, 0.5, 1\}$ and $\delta = \frac{1}{n^2}$. We evaluate public-data-assisted algorithms with the following public datasets: **Left:** 2018 California (CA-18); **Center:** 2010 Pennsylvania (PA-10); **Right:** 2018 Ohio (PA-10). The *x-axis* uses a logarithmic scale. Results are averaged over 5 runs, and error bars represent one standard error.

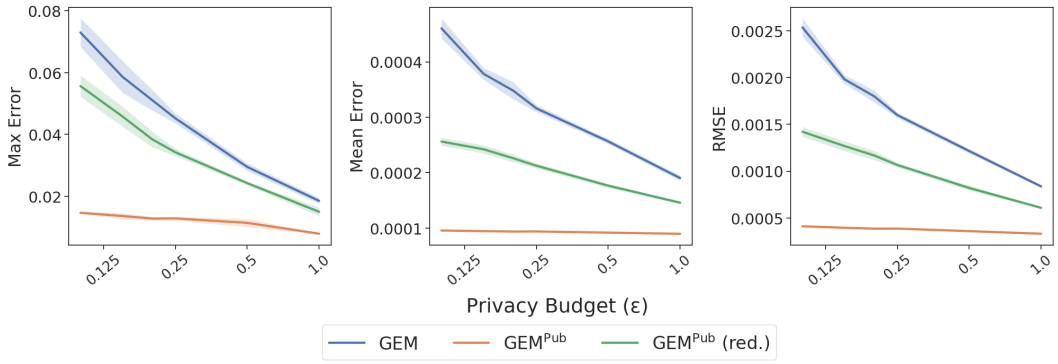


Figure 5: Max, mean, and mean squared error for 3-way marginals on ADULT (workloads = 286) with privacy budgets $\varepsilon \in \{0.1, 0.15, 0.2, 0.25, 0.5, 1\}$ and $\delta = \frac{1}{n^2}$. We evaluate GEM using both the complete public data (GEM$^{\text{Pub}}$) and a reduced version that has fewer attributes (GEM$^{\text{Pub}}$ (reduced)). The *x-axis* uses a logarithmic scale. Results are averaged over 5 runs, and error bars represent one standard error.

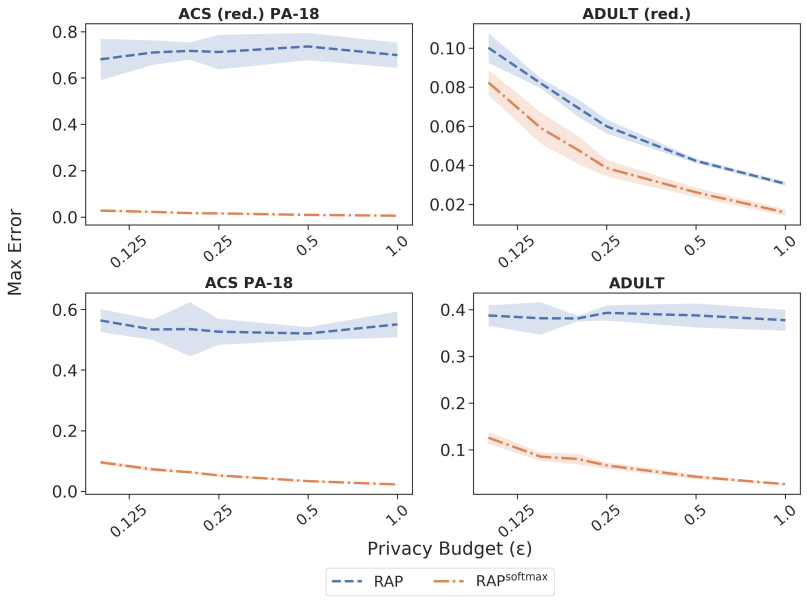


Figure 6: Comparison of RAP and RAP$^{\text{softmax}}$ w.r.t max error for 3-way marginals evaluated on ADULT and ACS PA-18 using privacy budgets $\varepsilon \in \{0.1, 0.15, 0.2, 0.25, 0.5, 1\}$ and $\delta = \frac{1}{n^2}$. The *x-axis* uses a logarithmic scale. We evaluate using the following workload sizes: ACS (reduced) PA-18: 455; ACS PA-18: 4096; ADULT (reduced): 286; ADULT: 35. Results are averaged over 5 runs, and error bars represent one standard error.

F.3 Comparisons against RAP

In Figure 6, we show failures cases for RAP. Again, we see that RAP$^{\text{softmax}}$ outperforms RAP in every setting. However, we observe that aside from ADULT (reduced), RAP performs extremely poorly across all privacy budgets.

To account for this observation, we hypothesize that by projecting each measurement to Aydore et al. [2]'s proposed continuous relaxation of the synthetic dataset domain, RAP produces a synthetic dataset that is inconsistent with the semantics of an actual dataset. Such inconsistencies make it more difficult for the algorithm to do well without seeing the majority of high error queries.

Consider this simple example comparing GEM and RAP. Suppose we have some binary attribute $A \in \{0, 1\}$ and we have $P(A = 0) = 0.2$ and $P(A = 1) = 0.8$. For simplicity, suppose that the initial answers at $t = 0$ for both algorithms is 0 for the queries $q_{A=0}$ and $q_{A=1}$. Assume at $t = 1$ that the privacy budget is large enough such that both algorithms select the max query $q_{A=1}$, which gives us an error or 0.8. After a single iteration, both algorithms can reduce the error of this query to 0. In RAP, the max error then is 0.2 (for the next largest error query $q_{A=0}$). However for GEM to get the correct answer for $q_{A=1}$, it must learn a distribution (due to the softmax activation function) such that $P(A = 1) = 0.8$, which naturally forces $P(A = 0) = 0.2$. In this way, GEM can reduce the errors of both queries in one step, giving it an advantage over RAP.

In general, algorithms within the Adaptive Measurements framework have this advantage in that the answers it provides must be consistent with the data domain. For example, if again we consider the two queries for attribute A, a simple method like the Gaussian or Laplace mechanism has a nonzero probability of outputting noisy answers for $q_{A=0}$ and $q_{A=1}$ such that $P(A = 0) + P(A = 1) \neq 1$. This outcome however will never occur in Adaptive Measurements.

Therefore, we hypothesize that RAP tends to do poorly as you increase the number of high error queries because the algorithm needs to select each high error query to obtain low error. Synthetic data generation algorithms can more efficiently make use of selected query measurements because their answers to all possible queries must be consistent. Referring to the above example again, there

may exist two high error queries $q_{A=0}$ and $q_{A=1}$, but only one needs to be sampled to reduce the errors of both.

We refer readers to Appendix F.6, where we use the above discussion to account for how the way in which the continuous attributes in ADULT are preprocessed can impact the effectiveness of RAP.

F.4 Discussion of HDMM

HDMM [14] is an algorithm designed to directly answer a set of workloads, rather than some arbitrary set of queries. In particular, HDMM optimizes some strategy matrix to represent each workload of queries that in theory, facilitates an accurate reconstruction of the workload answers while decreasing the sensitivity of the privacy mechanisms itself. In their experiments, McKenna et al. [14] show strong results w.r.t. RMSE, and the U.S. Census Bureau itself has incorporated aspects of the algorithm into its own releases [12].

We originally planned to run HDMM as a baseline for our algorithms in the standard setting, but after discussing with the original authors, we learned that currently, the available code for HDMM makes running the algorithm difficult for the ACS and ADULT datasets. There is no way to solve the least square problem described in the paper for domain sizes larger than 10^9, and while the authors admit that HDMM could possibly be modified to use local least squares for general workloads (outside of those defined in their codebase), this work is not expected to be completed in the near future.

We also considered running HDMM+PGM [15], which replaces the least squares estimation problem a graphical model estimation algorithm. Specifically, using (differentially private) measurements to some set of input queries, HDMM+PGM infers answers for any workload of queries. However, the memory requirements of the algorithm scale exponentially with dimension of the maximal clique of the measurements, prompting users to carefully select measurements that help build a useful junction tree that is not too dense. Therefore, the choice of measurements and cliques can be seen as hyperparameters for HDMM+PGM, but as the authors pointed out to us, how such measurements should be selected is an open problem that hasn't been solved yet. In general, cliques should be selected to capture correlated attributes without making the size of the graphical model intractable. However, we were unsuccessful in finding a set of measurements that achieved sensible results (possibly due to the large number of workloads our experiments are designed to answer) and decided stop pursuing this endeavor due to the heavy computational resources required to run HDMM+PGM. We leave finding a proper set of measurements for ADULT and ACS PA-18 as an open problem.

F.5 Effectiveness of optimizing over past queries

One important part of the adaptive framework is that it encompasses algorithms whose update step uses measurements from past iterations. In Figure 7, we verify claims from Hardt et al. [11] and Liu et al. [13] that empirically, we can significantly improve over the performance of MWEM when incorporating past measurements.

F.6 Evaluating on ADULT* and LOANS

In Figure 8, we reproduce the experiments on the ADULT (which we denote as ADULT*) and LOANS datasets presented in Aydore et al. [2]. Like Aydore et al. [2], we obtain the datasets from `https://github.com/giusevtr/fem`. Empirically, we find that GEM outperforms all baseline methods. In addition, while RAP performs reasonably well, we observe that by confining $\mathcal{D}$ to $\left\{\sigma(M)|M \in \mathbb{R}^{n' \times d}\right\}$ with the softmax function, $\text{RAP}^{\text{softmax}}$ performs better across all privacy budgets.

To account for why RAP performs reasonably well with respect to max error on ADULT* and LOANs but very poorly on ADULT and ACS, we refer back to our discussion about the issues of RAP presented in Appendix F.3 in which argue that by outputting synthetic data that is inconsistent with any real dataset, RAP performs poorly when there are many higher error queries. ADULT* and LOANs are preprocessed in a way such that continous attributes are converted into categorical (technically ordinal) attributes, where a separate categorical value is created for each unique value that the continuous attribute takes on in the dataset (up to a maximum of 100 unique values). When processed in this way, k-way marginal query answers are sparser, even when k is relatively small

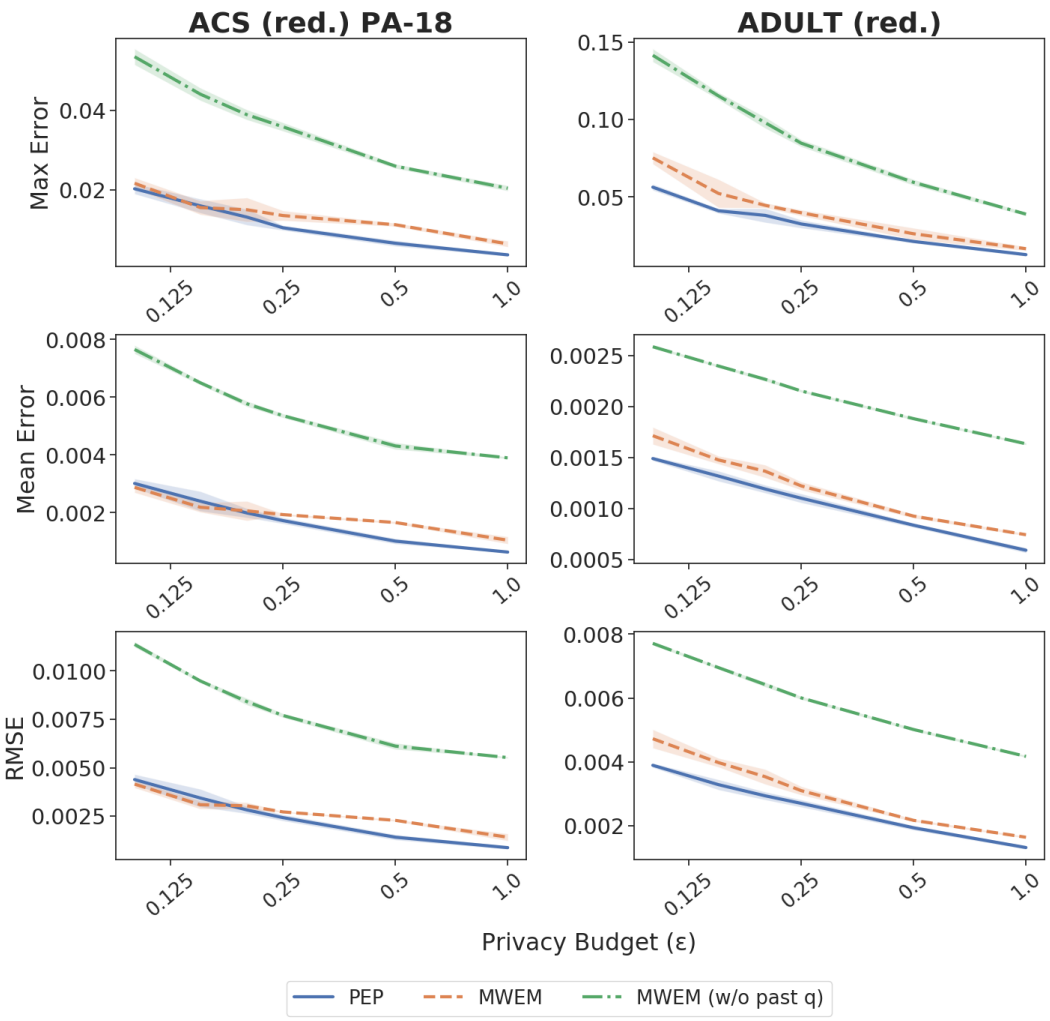


Figure 7: Comparison of max, mean, and root mean squared errors against vanilla MWEM that does not use queries sampled during past iterations, evaluated on 3-way marginals with privacy budgets $\varepsilon \in \{0.1, 0.15, 0.2, 0.25, 0.5, 1\}$ and $\delta = \frac{1}{n^2}$. The *x-axis* uses a logarithmic scale. Results are averaged over 5 runs, and error bars represent one standard error.

(≤ 5). However, Liu et al. [13] preprocess continuous variables in the ADULT and ACS dataset by constructing bins, resulting in higher error queries.

For example, suppose in an unprocessed dataset (with n rows), you have 3 rows where an attribute (such as income) takes on the values $16,587$, $15,984$, and $18,200$. Next, suppose there exists datasets A and B, where dataset A maps each unique value to its own category, while dataset B constructs a bin for values between $15,000$ and $20,000$. Then considering all 1-way marginal queries involving this attribute, dataset A would have 3 different queries, each with answer $\frac{1}{N}$. Dataset B however would only have a single query whose answer is $\frac{3}{N}$. Whether a dataset should be preprocessed as dataset A or dataset B depends on the problem setting.[3] However, this (somewhat contrived) example demonstrates how dataset B would have more queries with high value answers (and therefore more queries with high initial errors, assuming that the algorithms in question initially outputs answers that are uniform/close to 0).

[3] Although we would argue that in many cases, dataset B makes more sense since it is more likely for someone to ask—"How many people make between $15,000$ and $20,000$ dollars?"—rather than—"How many people make $15,984$ dollars?".

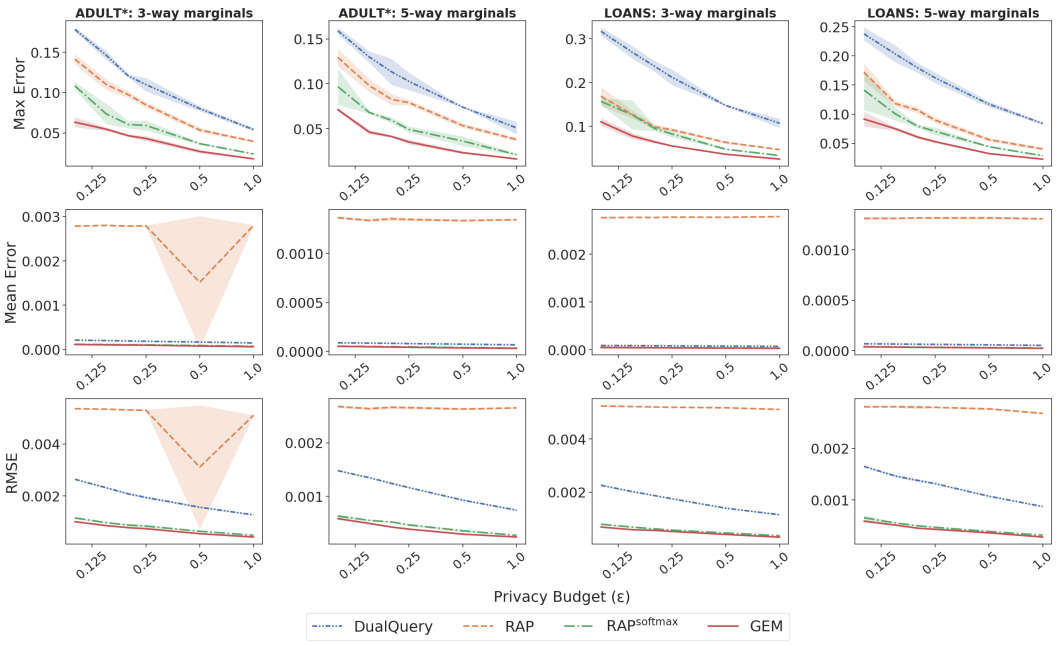


Figure 8: Max, mean, and root mean squared errors for 3-way marginals with a workload size of 64. Methods are evaluated on ADULT* and LOANS datasets using privacy budgets $\varepsilon \in \{0.1, 0.15, 0.2, 0.25, 0.5, 1\}$ and $\delta = \frac{1}{n^2}$. The *x-axis* uses a logarithmic scale. Results are averaged over 5 runs, and error bars represent one standard error.

In our experiments with 3-way marginal queries, ADULT (where workload is 286) and ADULT* (where the workload is 64) have roughly the same number queries (334, 128 vs. 458, 996 respectively). However, ADULT has 487 queries with answers above 0.1 while ADULT* only has 71. Looking up the number of queries with answers above 0.2, we count 181 for ADULT and only 28 for ADULT*. Therefore, experiments on ADULT* have fewer queries that RAP needs to optimize over to achieve low max error, which we argue accounts for the differences in performance on the two datasets.

Finally, we note that in Figure 6, RAP has relatively high mean error and RMSE. We hypothesize that again, because only the queries selected on each round are optimized and all other query answers need not be consistent with the optimized ones, RAP will not perform well on any metric that is evaluated over all queries (since due to privacy budget constraints, most queries/measurements are never seen by the algorithm). We leave further investigation on how RAP operates in different settings to future work.