DP-GMMs: Differentially Private Gaussian Mixture Models

Charlie Harrington
Institute for Applied Computational Science
Harvard University
150 Western Ave., Boston, MA 02134
charlesharrington@g.harvard.edu

Jeffrey Mayolo
Institute for Applied Computational Science
Harvard University
150 Western Ave., Boston, MA 02134
jeffmayolo@g.harvard.edu

Abstract

A common problem faced in the field of data science is how to build models for underdetermined problems. Gaussian Mixture Models are a frequently used tool to address this issue that provide an approximated probability density function for the given data. When this dataset is sensitive information, it is important to create these mixture models in a privacy preserving manner. In this paper, we outline, implement, and compare existing methods for developing differentially private Gaussian Mixture Models. In particular, we construct GMMs on both simulated and real datasets using Local-DP and Central-DP frameworks and compare the methods' performance in terms of privacy and utility.

1 Motivation

Throughout this semester, we have examined various forms of differential privacy. The focus of these methods has been how to release information about the data in a way that provides a reasonable guarantee of privacy for the individuals in the dataset. This released information has been static. Whether it be summary statistics, histograms, or gradients, these releases have been deterministic. There are many purposes, however, for which stochastic representations of sensitive data could be of great value. In many underdetermined problems, having functional representations of data provides the opportunity for analysis and inference that could not be accomplished otherwise. There are numerous ways to construct such functional representations: parametric and non-parametric density estimation, kernel density estimation (KDE), and Gaussian Mixture Models (GMMs), to name a few. In this paper, we will focus on GMMs and how to construct them from sensitive data in a differentially private manner.

1.1 Introduction

Differentially-private mechanisms have become the gold standard for the release of statistical data in a manner that protects the individuals in the dataset. The focus of these methods has been how to release information about the data in a way that provides a reasonable guarantee of privacy for the individuals in the dataset. The field of differential privacy is constantly growing with new methods being released each year. Nonetheless, much research is still yet to be done for privatized non-deterministic outputs. GMMs behave as a hybrid between clustering and density estimators that have large variety of practical uses, such as the one outlined above. A GMM is comprised of $k \in \{1, \ldots, K\}$ Gaussians, K being the number of clusters in the data [1]. Whereas KDE assigns each datapoint its own Gaussian component, resulting in a non-parametric estimation, GMM assigns a cluster of points to a single component. Each component has a mean μ, a covariance Σ, and a mixing probability π that defines how big or small the Gaussian will be. The sum of all mixing probabilities is equal to one, $\sum_k \pi_k = 1$. Building GMMs from raw data is an iterative process that utilizes the expectation-maximization (EM) algorithm [2]. The result is a non-parametric representation of the data distribution that can then be used for a number of further applications.

In many instances, GMMs are used on datasets consisting of sensitive information. Questions posed by the social sciences are often inherently underdetermined. Evaluating the impact of environmental factors on human performance are an example of such a problem in which the observed covariates can be mapped to observed outcomes that can then

be used to predict performance. These observed covariates can be sensitive data, and thus it is important to not only build a GMM that allows for analysis of the underdetermined problem, but to do so in a way that preserves privacy.

1.2 Research Question

This led us to our initial research question: How can we build differentially private Gaussian Mixture Models in a manner? Through our investigation of the existing literature, we discovered that there has already been a considerable amount of work on this topic, as Section 2 will discuss. There are different approaches to constructions DP-GMMs whether one adopts a Local-DP or Central-DP framework. Because this choice depends on where the barrier of trust is deemed to be most appropriate, both approaches are worth attention. The contribution of this work is to compare the two DP paradigms by implementing them and evaluating the trade-off between privacy and utility of the methods.

2 Related Work

2.1 Local-DP

Local differential privacy maintains the least privacy assumptions through the removal of a trusted curator from the mechanism. Locally private mechanisms add noise to the data in the collection step. This early noise addition guarantees that nobody will see your data getting rid of the trust barrier that exists in a central model between the user and the person implementing the differentially private mechanism.

There has been much research in the area of how to best add noise to the data for different private outputs. The most commonly used form of local differential privacy is through randomized response [3]. The randomized response mechanism works by estimating proportions of a population through the "coinflip" method. This can be applied to histograms by approximating the proportion of the population in each bin of the histogram.

However, there are many alternative methods for local DP histograms that are proposed to work better than randomized response. One example is randomized aggregatable privacy-preserving ordinal response (RAPPOR) [4]. The RAPPOR mechanism treats each person's data as bit strings and uses bloom filters on those strings to protect the privacy of the individuals. This allows for a broad band of crowd sourcing statistics that can be used to view and summarize the data.

Another common method for locally private distribution estimation is through unary encoding [5]. This method one-hot-encodes a response $v \in d$ as a binary vector. It then generates a perturbed encoding by flipping the response according to some probability p and q. It then aggregates the response to create a vector of approximate counts.

Additional research has been done to expand on these methods as well such as the development of $k - ary$ randomized response [6]. Nonetheless, for the purposes of this paper we will focus on the use of traditional randomized response to build our distributions. Randomized response was the most densely covered in class and will allow for us to ensure accuracy and consistency in our results. Additionally, the data analyzed and simulations run were do not require heavy computation so a computationally more efficient method was not required. A deeper analysis of the methodology of randomized response will be provided in Section 3.2.

2.2 Central-DP

There has been considerable work done related to building GMMs under the central model of differential privacy. One approach is to ensure that the underlying algorithm is executed in a differentially private manner, this being expectation maximization (EM). One of the major challenges of this is allocating the privacy for an iterative algorithm; each step incurs a privacy cost. To address this, Park et al. propose bounding the privacy cost through the combined use of the moments accountant (MA) [7] and zero-mean concentrated differential privacy (zCDP) [8]. While the paper provides empirical results for the DP-EM algorithm in building Gaussian mixture models, the authors indicate the general usability of the algorithm across its variety of uses [9].

Other literature narrows in on the iterative steps of GMM development and adding noise to the learned component parameters at each step [10, 11]. The method proposed by Wu et al. is intuitively straight-forward: at each iteration update the learned K-tuple parameters π, μ, and σ, add noise, and post-process to maintain parameter intrinsic characteristics. The Laplace mechanism is used to add noise and incorporated the global sensitivity of each parameter.

These sensitivities are

$$GS_\pi = \frac{K}{n} \tag{1}$$

$$GS_\mu = \frac{4RK}{n} \tag{2}$$

$$GS_\Sigma = \frac{12nKR^2 + 8K^2R^2}{n^2} \tag{3}$$

where n and K are the number of datapoint and Gaussian components, respectively, and R is the upper bound of the L_1-norm of the data. The global privacy budget ϵ is allocated as

$$\epsilon_I = 0.1\epsilon \tag{4}$$

$$\epsilon_\pi = \frac{0.04\epsilon}{T} \tag{5}$$

$$\epsilon_\mu = \frac{0.16\epsilon}{TK} \tag{6}$$

$$\epsilon_\Sigma = \frac{0.7\epsilon}{TK} \tag{7}$$

where T is the maximum number of iterations and ϵ_I is the privacy budget for the parameter initialization. The authors use PrivGene [12] for this initialization. As both sensitivities and privacy budgets show, the noise scales with n and T, and the authors discuss the issues related to these hyperparameters. After the noise is added, the mixture weights need to be normalized to ensure $\sum_k \pi_k = 1$ and the covariance matrix needs to be adjusted to ensure it is positive semi-definite. Kamath et al. deal with building GMMs from high-dimensional data with their approach of using principal component analysis to project the data into a low-dimensional space, then recursively clustering within that space.

As [10, 11] allude to, GMMs can be broken into to parts from a top-level view: clustering and kernel density estimation within each cluster. The DP literature in these two areas is rich [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]. The sources all demonstrate methods for building GMMs by completing the two main steps while ensuring differential privacy.

3 Methodology

3.1 Overview

Our overall methodology for this paper is as follows. It is important to note that all implementation and experimentation will be conducted on single-dimension datasets. We will first implement ways to create GMM using both Local- and Central-DP, as Sections 3.2 and 3.3 will outline[1]. To provide a proof of concept, we'll demonstrate these implementations on a small dataset that exhibits behaviors of a multi-Gaussian distribution in Section 4.1. We'll then compare the utility of our implementations with simulated data in Section 4.2.

3.2 Local-DP

As stated in previous sections, the locally private GMM will be created utilizing the randomized response mechanism. The randomized response mechanism is designed to estimate a counting query. This can be applied to create a histogram by binning the data and applying the mechanism to each bin in the histogram. The query provides the following response for $y \in \{-1, 1\}$.

$$Q(y) = \begin{cases} +y, & \text{w.p. } \frac{e^\epsilon}{e^\epsilon+1} \\ -y, & \text{w.p. } \frac{1}{e^\epsilon+1} \end{cases} \tag{8}$$

Running this mechanism returns an array of -1's and 1's for each bin of the histogram where the means of each array is proportional to the proportion that each bin is. These values are corrected using the following correction factor:

$$c_\epsilon = \frac{e^\epsilon + 1}{e^\epsilon - 1} \text{ and } E(c_\epsilon \times Q(y)) = y \tag{9}$$

This mechanism for estimating proportions is ϵ-DP. However, this mechanism only estimated the proportion size of each bin. In order to be used in a GMM it must be converted back to a full distribution. It should be noted that this

[1]The code for these implementations is available at `https://github.com/charlieh205/DP-GMM`.

method for approximating DP counting queries returns negative values. Since a distribution cannot be created with negative frequencies, the data is clipped to 0. It should be noted that the clipping to 0 adds an additional source of bias as the proportions previously summed to 1 and do not with with negative proportions being raised to 0. This results in a distribution with a higher number of observations then the original dataset. Nonetheless, since a GMM produces estimates of a mean, proportion, and standard deviation, the higher sample size will not effect the results.

Once the data is clipped to 0, the proportions are multiplied by n where n is the number of observations in the original dataset. This provides us with the number of observations in each bin. These values can then be used with the value of each bin to recreate the distribution in. a differentially private manner. Once this process of distribution recreation is complete the data can be utilized using Scikit-learn's `GaussianMixture` object to fit and plot the distributions [25]. With this, we implemented our LDP-GMM as shown in Alg. 1.

Algorithm 1 `LDPGMM`: Differentially private Gaussian mixture model under local model

Input: X, K, n, ϵ
Output: $\Theta = (\pi, \mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K)$

bins $= [\min(X), \ldots, \max(X)]$ ▷ Create bins based on minimum and maximum of dataset
for bin in bins **do**
  for x in bin **do**
    $$Q(x) = \begin{cases} +x, & \text{w.p. } \frac{e^\epsilon}{e^\epsilon+1} \\ -x, & \text{w.p. } \frac{1}{e^\epsilon+1} \end{cases}$$
  end for
end for
$c = \frac{e^{\epsilon/2}+1}{e^\epsilon-1}$
$X' \leftarrow c \cdot Q$
$\Theta \leftarrow \text{GMM}(X', K)$ ▷ Create GMM using Scikit-learn [25]
return Θ

3.3 Central-DP

The majority of the works shown in Section 2.2 deal with multi-dimensional data. This is why expectation maximization is used – it provides an optimal solution to a non-convex problem by finding the best parameters for a higher-dimensional latent space. For data of a single dimension, we do not need to use the EM algorithm. Instead, we can use k-means clustering to cluster the data into K clusters, then determine the mean and variance for each of the clusters. Similarly, to create a Central-DP GMM, we only need to cluster the data in a differentially private manner, calculate the weights, means, and variances of the DP clusters, then add noise to these values. We'll use the Laplace mechanism for this noise addition.

We split the privacy cost among the sub-steps in a very similar manner to that of [9].

$$\epsilon_I = 0.1\epsilon \tag{10}$$
$$\epsilon_\pi = 0.04\epsilon \tag{11}$$
$$\epsilon_\mu = \frac{0.16\epsilon}{K} \tag{12}$$
$$\epsilon_\Sigma = \frac{0.7\epsilon}{K} \tag{13}$$

To cluster the data, we use Diffprivlib's `KMeans` class [26]. The last piece of information needed is the global sensitivities of π, μ, Σ. We'll consider a dataset $X \in [a, b]$. For the mixture weights, the worst case is if a single datapoint is changed so that it goes from one cluster to another. Thus, the global sensitivity is

$$GS_\pi = \frac{2}{n}. \tag{14}$$

For both μ and π, we'll consider the case when $x_i = a \; \forall x_i \in X$ and X' has one datapoint $x_i = b$. The global sensitivities of π and Σ are then

$$GS_\mu = \frac{b-a}{n} \tag{15}$$
$$GS_\Sigma = \frac{(n-1)(a-b)^2}{n^2} \tag{16}$$

It is worth noting that Eq. (16) is only true under the condition that X is a one-dimensional dataset. With this, we implemented our CDP-GMM as shown in Alg. 2.

Algorithm 2 `CDPGMM`: Differentially private Gaussian mixture model under central model

Input: X, K, n, ϵ
Output: $\Theta = (\pi, \mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K)$

$\epsilon_I = 0.1\epsilon, \epsilon_\pi = 0.04\epsilon, \epsilon_\mu = \frac{0.16\epsilon}{K}, \epsilon_\Sigma = \frac{0.7\epsilon}{K}$ ▷ Allocate privacy budget
$GS_\pi = \frac{2}{n}, GS_\mu = \frac{b-a}{n}, GS_\Sigma = \frac{(n-1)(a-b)^2}{n^2}$ ▷ Calculate sensitivities
`KMeans` (X, ϵ_I) ▷ Cluster data using Diffprivlib [26]
for $k = 1$ to K **do**
    $\pi_k \leftarrow \frac{n_k}{n} + \text{Lap}\left(\frac{GS_\pi}{\epsilon_\pi}\right)$
    $\mu_k \leftarrow \frac{1}{n_k}\sum X_k + \text{Lap}\left(\frac{GS_\mu}{\epsilon_\mu}\right)$
    $\Sigma_k \leftarrow \frac{1}{n_k}\sum X_k^2 - \left(\frac{1}{n_k}\sum X_k\right)^2 + \text{Lap}\left(\frac{GS_\Sigma}{\epsilon_\Sigma}\right)$
end for
return Θ

4 Results

4.1 Estimating Old Faithful Data Distribution

To demonstrate proof of concept of our methodology, we utilized data from the eruption wait times from Yellowstone's Old Faithful Geyser. This dataset was used as it is a commonly known and tested mixture of Gaussian's. Figure 1 shows the distribution of wait times between eruptions.

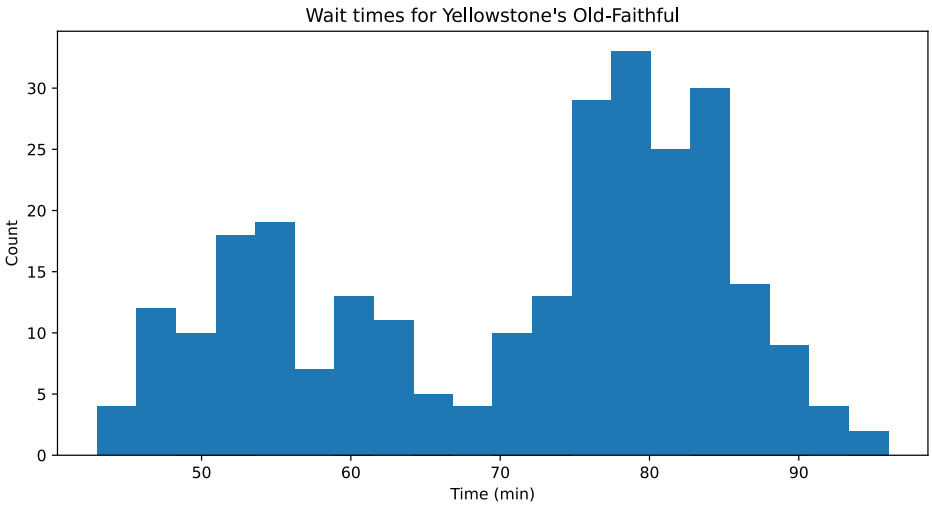

Figure 1: Old Faithful Dataset

We constructed three GMMs for this dataset: one with no differential privacy, one using Local-DP, and one using Central-DP. The resulting learned parameters are shown below in Table 1.

Table 1: Learned parameters for Old Faithful GMMs ($\epsilon = 3$).

	Component 1			Component 2		
	GMM	LDP-GMM	CDP-GMM	GMM	LDP-GMM	CDP-GMM
Mixing weight π	0.363	0.411	0.429	0.637	0.589	0.571
Mean μ	54.7	58.065	56.766	80.145	81.711	80.207
Variance Σ	35.318	89.394	34.509	33.79	73.93	32.916

We see that the mixing weights and means vary only slightly, more so for the LDP-GMM than CDP-GMM, but the variances for the LDP-GMM components are much larger than that of the other two models. To visualize these results, see Figure 2. It is very important to note that we cannot reach any definitive conclusions about the performance of the local model versus that of the central model based on this single example. Figure 2 serves merely as a proof-of-concept to demonstrate that our implementations of the local and central models are working as expected. To evaluate the performance of each method, we'll conduct some more experimentation.

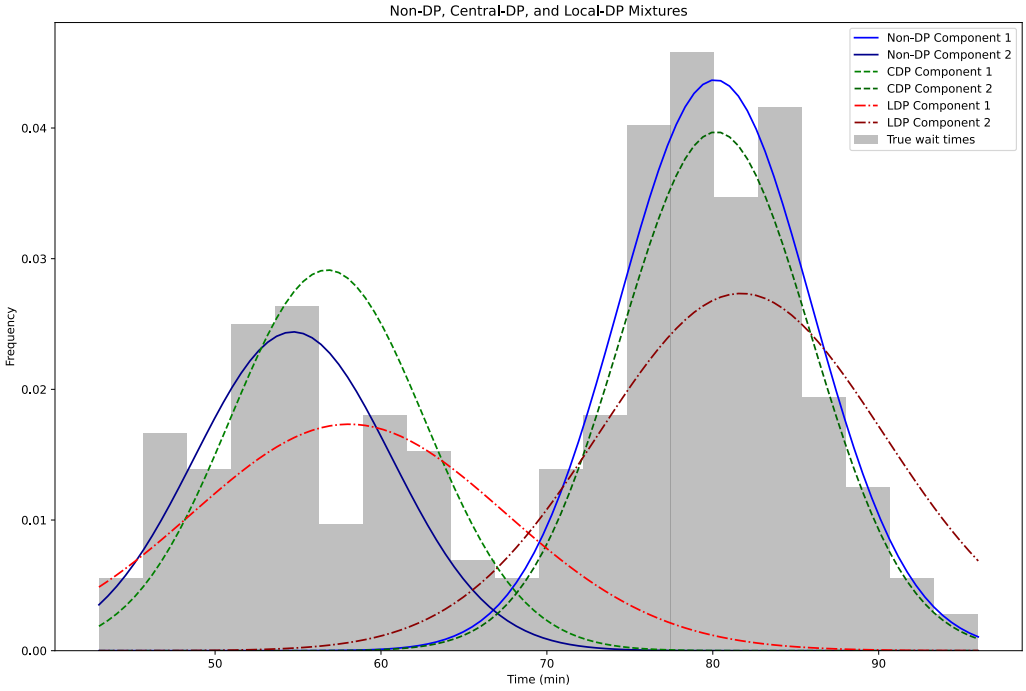

Figure 2: GMMs for Old Faithful Data

4.2 Simulated Data from Gaussian Kernels

4.2.1 RMSE over Sample Size

While analysis using the Old Faithful data set is effective for showing that the mechanisms works, it's important to analyze how the mechanism handles mixtures of multiple Gaussians as well as how it is effected by factors such as sample size. To do this we conducted a simulation varying the sample size from 100 to 1000. For each sample size we constructed a distribution that is a mixture of three Gaussians using the following parameters shown in Table 2.

Table 2: Parameters used to generate data.

	Component 1	Component 2	Component 3
π	0.15	0.35	0.5
μ	17	30	50
Σ	9	16	25

Using the generated data, we created twenty standard GMMs, twenty LDP-GMMs, and twenty CDP-GMMs (using $\epsilon = 3$). From the resulting parameters, we then calculated the root-mean squared error as compared to the true parameters of Table 2. We averaged these RMSE values across components and plotted them over the sample sizes, shown in Figure 3

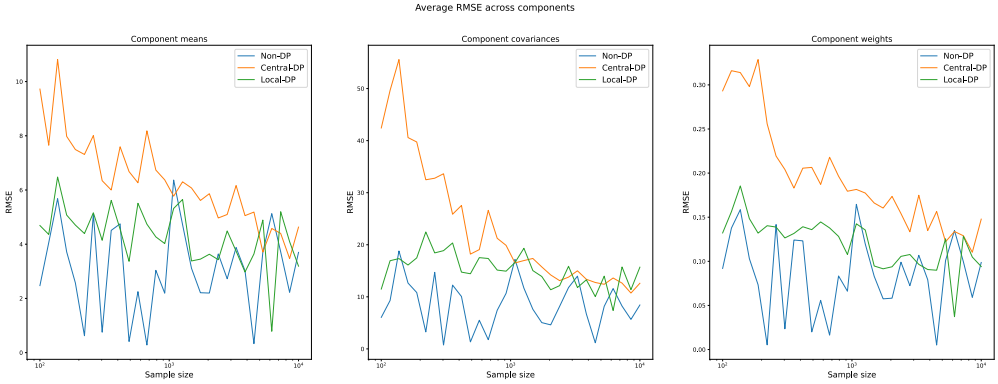

Figure 3: Average RMSE values for GMMs over ranging sample size

As we can see from the plots, it appears that the local model actually does a better job in general at estimating the component parameters in a differentially private manner. Once the sample size reaches a certain point, the two models reach near parity. As expected, we see that increasing the sample size does decrease the error for both the local and central models. This is because as sample size increases, the amount of noise added decreases.

4.2.2 RMSE over Epsilon

Using the same parameters as outlined in Table 2, we conducted a second simulation to compare the performance of the local and central models. Instead of varying sample size, however, we used different values of ϵ from $[0.01, 10]$ and calculated RMSE across the components in the same manner.

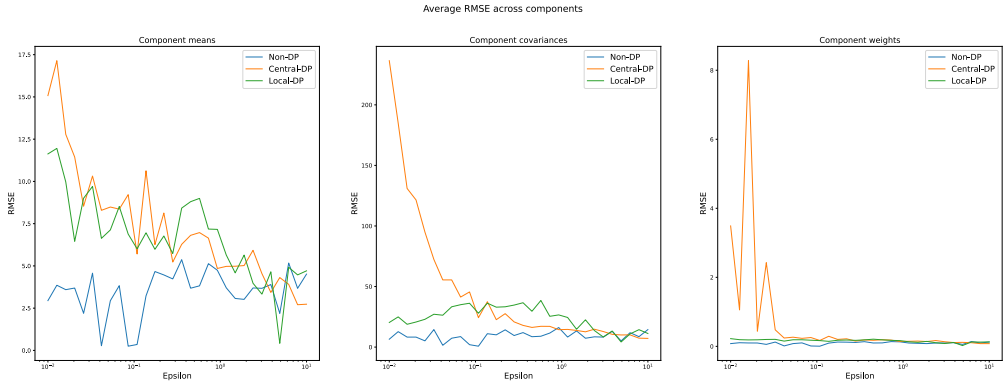

Figure 4: Average RMSE values for GMMs over ranging ϵ

The results of this simulation echo what we found in Section 4.2.1. In general, the parameters of the local model are more accurate than that of the central model, but the two achieve parity after ϵ reaches a certain point. Similar to increasing sample size, we also see that increasing ϵ reduces the error for both models, again because this corresponds to smaller noise addition.

4.2.3 Discussion

We saw that, in general, the local model is more accurate than the central model across sample size and ϵ. When considering reasons for this, it is important to think about what both randomized response and Gaussian Mixture models are doing to the dataset. To begin, GMMs are used to capture the overall behaviors of the data and its underlying unknown distribution. The fundamental assumption of GMMs is that this unknown distribution can be approximated with a mixture of Gaussian components. Thus, if two different datasets come from identical distributions, then in theory, the two GMMs should be similar. When creating a locally-DP histogram, randomized response attempts to recreate the general trends and behaviors of the original data. As such, it is not unreasonable to expect the non-DP GMM and the local-DP GMM to be relatively similar. We see in Figures 3 and 4 that this does, in fact, appear to be the case; the RMSE of the non-DP and local-DP GMMs are fairly similar for both experiments. With the central model, on the other hand, noise is being added directly to the parameters. The impact of this noise to the parameters on the final GMM is likely greater than that of noise applied to the dataset.

5 Limitations and Future Work

There are a few key limitations to our current implementations, the first begin the usability of the local model. As previously mention, we constrained the scope of this project to one-dimensional datasets. The result of this decision is that our implementations do not scale to higher dimensional data. As outlined in Section 3.2, we utilized randomized response for our local model. Randomized response estimates the proportion of the total dataset that each value has. This means that the data must be put into bins, which has two potential implications. The first is that only integer-valued datasets can be used. This is true for the results seen in Section 4; both the Old Faithful and simulated datasets were integer-valued. If continuous data was used, the data would be binned to the nearest integer. Because GMM is built off of data reconstructed from this binning, this means that information about the variances would be lost from rounding.

The way in which we employed our models for both the Old Faithful data as well as the simulated data has the potential for privacy leakage. This is due to the bounds of the dataset being determined from the data itself. Warnings will arise when this occurs, informing the user of the potential privacy leakage. To avoid this, inputting bounds based on reasonable information about the dataset can be used.

An area for future work is to expand to higher-dimensional datasets. Doing this would require new implementations for both the local and central models and would include researching further into the literature on local differential privacy for high-dimensional data.

Other future work includes evaluating model performance of the models for use-cases in which the dataset is not clearly composed of Gaussian kernels. One of the primary reasons why GMMs are used in practice is because of the assumption that they can approximate any given distribution relatively well. It would be worth investigating how well DP-GMMs satisfy this assumption. The methods for evaluating performance used in this paper would need to be redefined.

6 Conclusion

Gaussian Mixture Models are a great solution for approximating data distributions from real-world problems. As more people and companies expect data privacy, it is important that these models are constructed in a differentially-private manner. In this paper, we have explored methods for doing this using both the Local-DP and Central-DP frameworks. We implemented one method from each framework for use on a one-dimensional dataset, saw that the Local model has better accuracy over a range of dataset sizes and values of ϵ, and discussed some potential reasons for these results. In reality, the choice between Local-DP and Central-DP frameworks does not necessarily depend on the accuracy of the models, but rather on where participants believe the barrier of trust should reside. We have demonstrated that both options have viable solutions with great opportunity for future exploration and development.

References

[1] Oscar Contreras Carrasco. Gaussian Mixture Models Explained, February 2020.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[3] Stanley L. Warner. Randomized response: a survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60 309:63–6, 1965.

[4] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 21st ACM Conference on Computer and Communications Security*, Scottsdale, Arizona, 2014.

[5] Tianhao Wang, Zitao Li, Ninghui Li, Milan Lopuhaä-Zwakenberg, and Boris Skoric. Locally Differentially Private Frequency Estimation with Consistency. *CoRR*, abs/1905.08320, 2019.

[6] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. Discrete Distribution Estimation under Local Privacy, 2016.

[7] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[8] Mark Bun and Thomas Steinke. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. *CoRR*, abs/1605.02065, 2016. arXiv: 1605.02065.

[9] Mijung Park, James Foulds, Kamalika Choudhary, and Max Welling. DP-EM: Differentially Private Expectation Maximization. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 896–904. PMLR, April 2017.

[10] Yuncheng Wu, Yao Wu, Hui Peng, Juru Zeng, Hong Chen, and Cuiping Li. Differentially private density estimation via Gaussian mixtures model. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pages 1–6, 2016.

[11] Gautam Kamath, Or Sheffet, Vikrant Singhal, and Jonathan Ullman. Differentially Private Algorithms for Learning Mixtures of Separated Gaussians. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[12] Jun Zhang, Xiaokui Xiao, Y. Yang, Zhenjie Zhang, and Marianne Winslett. PrivGene: differentially private model fitting using genetic algorithms. In *SIGMOD '13*, 2013.

[13] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth Sensitivity and Sampling in Private Data Analysis. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 75–84, New York, NY, USA, 2007. Association for Computing Machinery. event-place: San Diego, California, USA.

[14] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially Private Combinatorial Optimization. In *SODA*. Society for Industrial and Applied Mathematics, January 2010. Edition: SODA.

[15] Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Locating a Small Cluster Privately. *arXiv e-prints*, page arXiv:1604.05590, April 2016. _eprint: 1604.05590.

[16] Kobbi Nissim and Uri Stemmer. Clustering Algorithms for the Centralized and Local Models. In Firdaus Janoos, Mehryar Mohri, and Karthik Sridharan, editors, *Proceedings of Algorithmic Learning Theory*, volume 83 of *Proceedings of Machine Learning Research*, pages 619–653. PMLR, April 2018.

[17] Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. Differentially Private Clustering in High-Dimensional Euclidean Spaces. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 322–331. PMLR, August 2017.

[18] Haim Kaplan and Uri Stemmer. Differentially Private k-Means with Constant Multiplicative Error. *CoRR*, abs/1804.08001, 2018. arXiv: 1804.08001.

[19] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially Private k-Means Clustering. *CoRR*, abs/1504.05998, 2015. arXiv: 1504.05998.

[20] Zhiyi Huang and Jinyan Liu. Optimal Differentially Private Algorithms for K-Means Clustering. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, SIGMOD/PODS '18, pages 395–408, New York, NY, USA, 2018. Association for Computing Machinery. event-place: Houston, TX, USA.

[21] Gautam Kamath, Jerry Li, Vikrant Singhal, and Jonathan R. Ullman. Privately Learning High-Dimensional Distributions. *CoRR*, abs/1805.00216, 2018. arXiv: 1805.00216.

[22] Sourav Biswas, Yihe Dong, Gautam Kamath, and Jonathan Ullman. CoinPress: Practical Private Mean and Covariance Estimation. In *Advances in Neural Information Processing Systems 33*. 2020.

[23] Ilias Diakonikolas, Moritz Hardt, and Ludwig Schmidt. Differentially Private Learning of Structured Discrete Distributions. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[24] Vishesh Karwa and Salil P. Vadhan. Finite Sample Differentially Private Confidence Intervals. *CoRR*, abs/1711.03908, 2017. arXiv: 1711.03908.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] Naoise Holohan, Stefano Braghin, Pól Mac Aonghusa, and Killian Levacher. Diffprivlib: the IBM differential privacy library. *ArXiv e-prints*, 1907.02444 [cs.CR], July 2019.

A Revision tracker

We will use this appendix to annotate any changes made from our original draft. The header will refer to the section that was modified and detail how it was changed.

A.1 Section 2.2

We clarified that the work of [9] did not create the moments account (MA) or zero-mean concentrated differential privacy (zCDP), but built on the works of [7] and [8].

A.2 Section 3.3

We added the conditions for which Eq. (16) is true.

A.3 Section 3.2

Here we added Alg. 1, the algorithm for our local-DP implementation.

A.4 Section 4.1

We added some commentary on the observed results and indicate that they are not a true indicator of overall performance.

A.5 Section 4.2.1

Here we added some brief comments on the trends observed using the Old Faithful dataset and how it cannot be used to evaluate the performance of the models overall.

A.6 Section 4.2.2

This section is entirely new. It shows our simulation results as we modify ϵ for a constant dataset size.

A.7 Section 4.2.3

This section is entirely new. Here we have further discussion of our results from Sections 4.2.1 and 4.2.2.

A.8 Section 5

We added some discussion of the limitations of our current implementations and updated the areas for future work.

A.9 Section 6

We updated our conclusion based on our new results.