Differentially Private Mean Embeddings with Random Features (DP-MERF) for Simple & Practical Synthetic Data Generation

Anonymous Authors[1]

Abstract

We present a differentially private data generation paradigm using random feature representations of kernel mean embeddings when comparing the distribution of true data with that of synthetic data. We exploit the random feature representations for two important benefits. First, we require a very low privacy cost for training deep generative models. This is because unlike kernel-based distance metrics that require computing the kernel matrix on all pairs of true and synthetic data points, we can detach the data-dependent term from the term solely dependent on synthetic data. Hence, we need to perturb the data-dependent term once-for-all and then use it until the end of the training. Second, we can obtain an *analytic* sensitivity of the kernel mean embedding as the random features are norm bounded by construction. This removes the necessity of hyper-parameter search for a clipping norm to handle the unknown sensitivity of an encoder network when dealing with high-dimensional data. We provide several variants of our algorithm, *differentially-private mean embeddings with random features* (DP-MERF) to generate (a) heterogeneous tabular data, (b) input features and corresponding labels jointly; and (c) high-dimensional image data. Our algorithm achieves better privacy-utility trade-offs than existing methods tested on several datasets.

1. Introduction

Classical approaches to differentially private (DP) data generation typically assumes a certain class of pre-specified queries. These DP algorithms produce a privacy-preserving synthetic database that is *similar* to the privacy-sensitive original data for that fixed query class (Mohammed et al., 2011; Xiao et al., 2010; Hardt et al., 2012; Zhu et al., 2017). However, specifying a query class upfront significantly limits the flexibility of the synthetic data, if data analysts hope to perform other machine learning tasks.

To overcome this inflexibility, many papers on DP data generation have utilized the recent advance in deep generative modeling. The majority of these approaches is based on the generative adversarial networks (GAN) (Goodfellow et al., 2014) framework, where a discriminator and a generator play a min-max form of game to optimize for the *Jensen-Shanon divergence* between the true and synthetic data distributions. The Jensen-Shanon divergence belongs to the family of divergence, known as *Ali-Silvey distance*, *Csiszár's ϕ-divergence* (Csiszr & Shields, 2004), defined as $D_\phi(\mathbb{P}, \mathbb{Q}) = \int_M \phi\left(\frac{\mathbb{P}}{\mathbb{Q}}\right) d\mathbb{Q}$ where M is a measurable space and P, Q are probability distributions. Depending on the form of ϕ, $D_\phi(\mathbb{P}, \mathbb{Q})$ recovers popular divergences[1] such as the Kullback-Liebler (KL) divergence ($\phi(t) = t \log t$). The GAN framework with the Jensen-Shanon divergence was also used for DP data generation (Park et al., 2018; Torkzadehmahani et al., 2019; Yoon et al., 2019).

Another popular family of distance measure is *integral probability metrics (IPMs)*, which is defined by $D(\mathbb{P}, \mathbb{Q}) = \sup_{f\in\mathcal{F}} \left|\int_M f d\mathbb{P} - \int_M f d\mathbb{Q}\right|$ where $\mathcal{F}$ is a class of real-valued bounded measurable functions on M. Depending on the class of functions, there are several popular choices of IPMs. For instance, when $\mathcal{F} = \{f : \|f\|_L \leq 1\}$, where $\|f\|_L := \sup\{|f(x) - f(y)|/\rho(x, y) : x \neq y \in M\}$ for a metric space (M, ρ), $D(\mathbb{P}, \mathbb{Q})$ yields the *Kantorovich* metric, and when M is separable, the Kantorovich metric recovers the *Wasserstein* distance, a popular choice for generative modelling such as Wasserstein-GAN and Wasserstein-VAE (Arjovsky et al., 2017; Tolstikhin et al., 2018). The GAN framework with the Wasserstein distance was also used for DP data generation (Xie et al., 2018; Frigerio et al., 2019).

As another example of IPMs, when $\mathcal{F} = \{f : \|f\|_\mathcal{H} \leq 1\}$, i.e., the function class is a unit ball in reproducing kernel Hilbert space (RKHS) associated with a positive-definite kernel k, $D(\mathbb{P}, \mathbb{Q})$ yields the *maximum mean discrepancy*

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]See Table 1 in (Nowozin et al., 2016) for various ϕ divergences in the context of GANs.

(MMD), $MMD(P, Q) = \sup_{f \in \mathcal{F}} \left| \int_M f d\mathbb{P} - \int_M f d\mathbb{Q} \right|$. In this case finding a supremum is analytically tractable and the solution is represented by the difference in the mean embeddings of each probability measure: $MMD(P, Q) = \|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathcal{H}}$, where $\mu_{\mathbb{P}} = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}}[k(\mathbf{x}, \cdot)]$ and $\mu_{\mathbb{Q}} = \mathbb{E}_{\mathbf{y}\sim\mathbb{Q}}[k(\mathbf{y}, \cdot)]$. For a characteristic kernel k, the squared MMD forms a metric, i.e., $MMD^2 = 0$, if and only if $P = Q$. MMD is also a popular choice for generative modelling in the GAN frameworks (Li et al., 2017a;b), as MMD compares two probability measures in terms of all possible moments (no information loss due to a selection of a certain set of moments); and the MMD estimator is in closed form (eq. 1) and easy to compute by the pair-wise evaluations of a kernel function using the points drawn from P and Q.

Here, we propose to use a particular form of MMD via *random Fourier feature* representations (Rahimi & Recht, 2008) of kernel mean embeddings for differentially private data generation. Our contributions are summarized below.

(1) We provide a simple, computationally efficient, and highly practical algorithm for DP data generation.

- *Simple:* This random feature representation of mean embedding (eq. 2) separates the mean embedding of the true data distribution (data-dependent) from that of the synthetic data distribution (data-independent). Hence, only the data-dependent term needs privatization. Random features provide an analytic sensitivity of the mean embedding, with which we simply adjust the noise level of a DP mechanism to produce DP data.
- *Computationally efficient:* As we have an analytic sensitivity, we do not need to search for the "right" clipping bound[2] which is necessary in many existing DP GAN-based methods. This reduces computational cost significantly, i.e., the computational cost of our method reduces to the usual SGD-based training of a generator.
- *Highly practical:* As the only term that needs privatization is simply the mean embedding of the true data distribution, we perturb the term once-for-all and then use it until the end of the training, resulting in a very low privacy loss for training deep generative models. Hence, our method achieves better privacy-utility trade-offs compared to existing GAN-based methods.

(2) Our algorithm accommodates several needs in privacy-preserving data generation.

- *Generating input and output pairs jointly*: We treat both input and output to be privacy-sensitive. This is different from the conditional-GAN type of methods.
- *Generating imbalanced and heterogeneous tabular data*: This is an extremely important condition for a DP method to be useful, as real world datasets frequently exhibit class-imbalance and heterogeneity.
- *Generating high-dimensional image data* using a low-dimensional-code based framework.

(3) We raise a question whether we really benefit from the DP versions of heavy machineary such as GAN and auto-encoder-based methods to generate the datasets that we typically consider in the DP literature.

- We consider 8 commonly-used tabular datasets and relatively simple image data (MNIST and FashionMNIST). For more complex data, it is necessary to use larger networks. However, the typical size of the classifiers in the DP literature todate is limited by 3-layer neural networks due to the challenge in finding a good privacy-utility trade-off. [3].
- Our vanilla method without the dimensionality reduction significantly outperforms other DP-GAN and our DP-auto-encoder-based methods for these data.
- As we are limited to these datasets and relatively small networks, we wonder if we truly benefit from the complicated-and-expensive-to-train GAN or auto-encoder type of DP data generation methods. If we can generate these datasets using much simpler methods like ours, the answer would be no.

We start by describing necessary background information before introducing our method.

2. Background

In the following, we describe the kernel mean embeddings with random features, and introduce differential privacy.

2.1. Random feature mean embeddings

Given the samples drawn from two probability distributions: $X_m = \{x_i\}_{i=1}^m \sim P$ and $X'_n = \{x'_i\}_{i=1}^n \sim Q$, the MMD estimator is defined as (Gretton et al., 2012):

$$\widehat{\mathrm{MMD}}^2(X_m, X'_n) = \frac{1}{m^2}\sum_{i,j=1}^m k(x_i, x_j) + \frac{1}{n^2}\sum_{i,j=1}^n k(x'_i, x'_j) - \frac{2}{mn}\sum_{i=1}^m\sum_{j=1}^n k(x_i, x'_j). \quad (1)$$

[2]Nobody reports how much computational power they used to find the right clipping norm in the existing DP GAN-based methods. From our experience, this step requires a significant amount of compute power as in each clipping norm candidate we need to train an entire generative model coupled with a discriminator.

[3]With access to public data the privacy-accuracy trade-off can be drastically improved, e.g., (Papernot et al., 2017).

The total computational cost of $\widehat{\mathrm{MMD}}(X_m, X'_n)$ is $O(mn)$, which is prohibitive for large-scale datasets.

A fast linear-time MMD estimator can be achieved by considering an approximation to the kernel function $k(x, x')$ with an inner product of finite dimensional feature vectors, i.e., $k(x, x') \approx \hat{\phi}(x)^\top \hat{\phi}(x')$ where $\hat{\phi}(x) \in \mathbb{R}^D$ and D is the number of features. The resulting MMD estimator is

$$\widehat{\mathrm{MMD}}^2_{rf}(P, Q) = \left\| \frac{1}{m} \sum_{i=1}^{m} \hat{\phi}(x_i) - \frac{1}{n} \sum_{i=1}^{n} \hat{\phi}(x'_i) \right\|_2^2, \quad (2)$$

which can be computed in $O(m + n)$, i.e., linear in the sample size. One popular approach to obtaining such $\hat{\phi}(\cdot)$ is based on random Fourier features (Rahimi & Recht, 2008) which can be applied to any translation invariant kernel, i.e., $k(x, x') = \tilde{k}(x - x')$ for some function $\tilde{k}$. According to Bochner's theorem (Rudin, 2013), $\tilde{k}$ can be written as $\tilde{k}(x - x') = \int e^{i\omega^\top (x - x')} \, \mathrm{d}\Lambda(\omega) = \mathbb{E}_{\omega \sim \Lambda} \cos(\omega^\top (x - x'))$, where $i = \sqrt{-1}$ and due to positive-definiteness of $\tilde{k}$, its Fourier transform Λ is nonnegative and can be treated as a probability measure. By drawing random frequencies $\{\omega_i\}_{i=1}^D \sim \Lambda$, where Λ depends on the kernel, $\tilde{k}(x - x')$ can be approximated with a Monte Carlo average. The vector of random Fourier features is given by

$$\hat{\boldsymbol{\phi}}(x) = (\hat{\phi}_1(x), \ldots, \hat{\phi}_D(x))^\top \quad (3)$$

where each coordinate is defined by

$$\begin{aligned} \hat{\phi}_j(x) &= \sqrt{2/D}\, \cos({\omega_j}^\top x), \\ \hat{\phi}_{j+D/2}(x) &= \sqrt{2/D}\, \sin(\omega_j^\top x), \end{aligned}$$

for $j = 1, \cdots, D/2$. The approximation error of these random features is studied in (Sutherland & Schneider, 2015).

2.2. Differential privacy

Given neighbouring datasets $\mathcal{D}$, $\mathcal{D}'$ differing by a single entry, a mechanism $\mathcal{M}$ is ϵ-DP if and only if $|L^{(o)}| \leq \epsilon, \forall o, \mathcal{D}, \mathcal{D}'$, where $L^{(o)}$ is the *privacy loss* of an outcome o defined by $L^{(o)} = \log \frac{Pr(\mathcal{M}(\mathcal{D})=o)}{Pr(\mathcal{M}(\mathcal{D}')=o)}$. A mechanism $\mathcal{M}$ is (ϵ, δ)-DP, if and only if $|L^{(o)}| \leq \epsilon$, with probability at least $1 - \delta$. DP guarantees a limited amount of information the algorithm reveals about any one individual. A DP algorithm adds randomness to the algorithms' outputs. Let a function $h : \mathcal{D} \mapsto \mathbb{R}^p$ computed on sensitive data $\mathcal{D}$ outputs a p-dimensional vector. We can add noise to h for privacy, where the level of noise is calibrated to the *global sensitivity* (Dwork et al., 2006), Δ_h, defined by the maximum difference in terms of L_2-norm $||h(\mathcal{D}) - h(\mathcal{D}')||_2$, for neighboring $\mathcal{D}$ and $\mathcal{D}'$ (i.e. differ by one data sample). The *Gaussian mechanism* that we will use in this paper outputs $\tilde{h}(\mathcal{D}) = h(\mathcal{D}) + \mathcal{N}(0, \sigma^2 \Delta_h^2 \mathbf{I}_p)$. The perturbed function $\tilde{h}(\mathcal{D})$ is (ϵ, δ)-DP, where σ is a function of ϵ, δ.

There are two important properties of DP. The *composability* theorem (Dwork et al., 2006) states that the strength of privacy guarantee degrades with repeated use of DP-algorithms. Furthermore, the *post-processing invariance* property (Dwork et al., 2006) tells us that the composition of any arbitrary data-independent mapping with an (ϵ, δ)-DP algorithm is also (ϵ, δ)-DP.

Differentially private stochastic gradient descent

Existing DP data generation algorithms under the GAN framework follow the two steps iteratively (Park et al., 2018; Torkzadehmahani et al., 2019; Xie et al., 2018; Frigerio et al., 2019). The discriminator is updated by *differentially private stochastic gradient descent* (DP-SGD) (Abadi et al., 2016), where the gradients computed on the data are altered by the Gaussian mechanism to limit the influence that each sample has on the model. The generator update is data-indepdent, as the generator update only requires accessing the privatized loss of the discriminator. Due to the post-processing invariance of DP, the resulting generator produces differentially private synthetic data. As DP-SGD requires accessing data numerously during training, a refined composition method to compute the cumulative privacy loss is proposed using the notion of Rényi Differential Privacy (RDP) (Mironov, 2017; Wang et al., 2019).

Definition 2.1 ((α, ϵ)-RDP)**.** *A mechanism is called ϵ Renyi differentially private with an order α if for all neighbouring datasets $\mathcal{D}, \mathcal{D}'$ the following holds:*

$$D_\alpha(\mathcal{M}(\mathcal{D}) || \mathcal{M}(\mathcal{D}')) \leq \epsilon(\alpha). \quad (4)$$

$D_\alpha(P||Q)$ is the α-Rényi divergence defined in Appendix. RDP takes an expectation over the outcomes of the DP mechanism, rather than taking a single worst case as in pure DP. Also, the RDP definition can benefit from the privacy amplification effect due to subsampling of data (See Theorem 9 (Wang et al., 2019)). A repeated use of RDP mechanisms composes by

Theorem 2.1 (Composition of RDP mechanisms)**.** *Let $f : \mathcal{D} \mapsto \mathcal{R}_1$ be (α, ϵ_1)-RDP. Let $g : \mathcal{R}_1 \times \mathcal{D} \mapsto \mathcal{R}_2$ be (α, ϵ_2)-RDP. Then, the mechanism releasing (X, Y), where $X \sim f(\mathcal{D})$ and $Y \sim g(\mathcal{D}, X)$ satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

Once the cumulative privacy loss using the RDP composition is computed, the RDP notion can be converted to the original definition of DP by the following proposition.

Proposition 1. *[From RDP to DP (Mironov, 2017) If $\mathcal{M}$ is a (α, ϵ)-RDP mechanism, then it also satisfies $\left(\epsilon + \frac{\log 1/\delta}{\alpha - 1}, \delta\right)$-DP for any $0 < \delta < 1$.*

The RDP-based composition yields a significantly smaller cumulative privacy loss than that by the linear sum of worst cases in the pure DP case.

3. Differentially private mean embeddings with random features (DP-MERF)

We first introduce the DP-MERF algorithm to learn the joint distribution over the input features $\mathbf{x}$ and output labels $\mathbf{y}$ (either categorical variables in classification, or continuous variables in regression). The benefit of learning the joint distribution is that we do not need to assume the information on the output labels to be public. By learning the joint distribution, we keep the ratio of the datapoints across different classes the same in the generated dataset as in the real dataset. This way our generated dataset is truthful to the privacy-sensitive original dataset in terms of both the distribution over the input features and the distribution over the labels.

3.1. DP-MERF for input/output pairs

Suppose a generator $G_{\boldsymbol{\theta}}$ (parameterized by $\boldsymbol{\theta}$) takes a pair of inputs $\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\mathbf{y}}$ drawn from a known distribution and outputs a pair of samples denoted by $\tilde{\mathbf{x}}_{\boldsymbol{\theta}}, \tilde{\mathbf{y}}_{\boldsymbol{\theta}} : G_{\boldsymbol{\theta}}(\mathbf{z}_{\mathbf{x}}, \mathbf{z}_{\mathbf{y}}) \mapsto \{\tilde{\mathbf{x}}_{\boldsymbol{\theta}}, \tilde{\mathbf{y}}_{\boldsymbol{\theta}}\}$. We consider the following objective function,

$$\widehat{\mathrm{MMD}}^2_{rf}(P_{\mathbf{x},\mathbf{y}}, Q_{\tilde{\mathbf{x}}_{\boldsymbol{\theta}},\tilde{\mathbf{y}}_{\boldsymbol{\theta}}}) = \left\| \widehat{\boldsymbol{\mu}}_{P_{\mathbf{x},\mathbf{y}}} - \widehat{\boldsymbol{\mu}}_{Q_{\mathbf{x},\mathbf{y}}} \right\|_F^2, \tag{5}$$

where F denotes the Frobenius norm. This type of joint maximum mean discrepancy was used in other papers (Zhang et al., 2019; Gao & Huang, 2018). The choice of kernel here is important, as now we want to compute the distance in terms of two types of inputs.

Here we consider a kernel from a product of two existing kernels, $k((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) = k_{\mathbf{x}}(\mathbf{x},\mathbf{x}')k_{\mathbf{y}}(\mathbf{y},\mathbf{y}')$, where $k_{\mathbf{x}}$ is a kernel for input features and $k_{\mathbf{y}}$ is a kernel for output. For regression, we could use the Gaussian kernel for both $k_{\mathbf{x}}$ and $k_{\mathbf{y}}$. For classification, we could use the Gaussian kernel for $k_{\mathbf{x}}$ and the polynomial kernel with order-1, $k_{\mathbf{y}}(\mathbf{y},\mathbf{y}') = \mathbf{y}^\top\mathbf{y}' + c$ for one-hot-encoded labels $\mathbf{y}$ and some constant c, for instance. In this case, the resulting kernel is also characteristic forming the corresponding MMD as a metric. See (Szabó & Sriperumbudur, 2018) for details.

We represent the mean embeddings using random features

$$\widehat{\boldsymbol{\mu}}_{P_{\mathbf{x},\mathbf{y}}} = \tfrac{1}{m}\sum_{i=1}^{m} \hat{\mathbf{f}}(\mathbf{x}_i, \mathbf{y}_i), \text{for true data} \tag{6}$$

$$\widehat{\boldsymbol{\mu}}_{Q_{\mathbf{x},\mathbf{y}}} = \tfrac{1}{n}\sum_{i=1}^{n} \hat{\mathbf{f}}(G_{\boldsymbol{\theta}}(\mathbf{z}_{\mathbf{x}_i}, \mathbf{z}_{\mathbf{y}_i})), \text{ for synthetic data}$$

where we define

$$\hat{\mathbf{f}}(\mathbf{x}_i, \mathbf{y}_i) := \mathrm{vec}(\hat{\boldsymbol{\phi}}(\mathbf{x}_i)\mathbf{f}(\mathbf{y}_i)^\top), \tag{7}$$

where $\mathbf{f}(\mathbf{y}_i) = \mathbf{y}_i$ for the order-1 polynomial kernel and $\mathbf{y}_i$ is one-hot-encoded. See Appendix for derivation. As a matrix notation, the random feature mean embedding in eq. 6 can be also written as

$$\widehat{\boldsymbol{\mu}}_{P_{\mathbf{x},\mathbf{y}}} = \begin{bmatrix}\mathbf{m}_1 & \cdots & \mathbf{m}_C\end{bmatrix} \in \mathbb{R}^{D\times C}$$

where the c'th column is defined by

$$\mathbf{m}_c = \frac{1}{m}\sum_{i\in c_c}^{m_c} \hat{\boldsymbol{\phi}}(\mathbf{x}_i) \tag{8}$$

where c_c is the set for the datapoints that belong to the class c, and m_c is the number of those datapoints. Recall D is the number of random features. C is the number of classes in the dataset. Notice that the sum in each column is over the number of instances that belong to the particular class c, while the divisor is the number of samples in the entire dataset, m. This brings difficulties in learning with this loss function when classes are highly imbalanced, as for rare classes m can be significantly larger than the sum of the corresponding column. Hence, for class-imbalanced datasets, we modify DP-MERF in eq. 6 with appropriately weighted one[4] where weights are denoted by $\{w_1, \cdots, w_C\}$

$$\widetilde{\boldsymbol{\mu}}_{P_{\mathbf{x},\mathbf{y}}} = \begin{bmatrix}\frac{1}{\omega_1}\mathbf{m}_1 & \cdots & \frac{1}{\omega_C}\mathbf{m}_C\end{bmatrix}$$

where the vector of weights is defined by

$$\boldsymbol{\omega} = [\omega_1, \cdots, \omega_C], \tag{9}$$

and $\omega_c = \frac{m_c}{m}$. By dividing by the weights, now each column has a similar order of strength regardless of the number of datapoints belonging to the specific class.

Here we privatize the weights and also the mean embedding of each column separately, using the two mechanisms defined below.

Definition 3.1 ($\mathcal{M}_{weights}$)**.** *The mechanism takes a dataset $\mathcal{D}$ and computes eq. 9. It outputs the privatized weights given a privacy parameter σ and the sensitivity $\Delta_{\boldsymbol{\omega}}$,*

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \mathcal{N}(0, \sigma^2(\Delta_{\boldsymbol{\omega}})^2\mathbf{I}_C), \tag{10}$$

where C is the number of classes.

Note that privatizing weight vector is analogous to privatizing the mixing coefficients in (Park et al., 2017). If there is one datapoint's difference in the neighbouring two datasets, only two elements can differ in the weight vector, resulting in the sensitivity of $\Delta_{\boldsymbol{\omega}} = \frac{\sqrt{2}}{m}$.

Definition 3.2 ($\mathcal{M}_{\mathbf{m}_c}$)**.** *The mechanism takes a dataset $\mathcal{D}$ and computes eq. 8. It outputs the privatized quantity given a privacy parameter σ and the sensitivity $\Delta_{\mathbf{m}_c}$,*

$$\tilde{\mathbf{m}}_c = \mathbf{m}_c + \mathcal{N}(0, \sigma^2(\Delta_{\mathbf{m}_c})^2\mathbf{I}_D) \tag{11}$$

where D is the number of random features.

[4] We arrive at this expression if we modify the kernel on the labels by a weighted one, i.e., $k_{\mathbf{y}}(\mathbf{y},\mathbf{y}') = \sum_{c=1}^{C}\frac{1}{\omega_c}\mathbf{y}_c{}^\top\mathbf{y}'_c$.

Algorithm 1 DP-MERF for generating input/output pairs

Require: Dataset $\mathcal{D}$, and a privacy level (ϵ, δ)

Ensure: (ϵ, δ)-DP input output samples for all classes

Step 1. Given (ϵ, δ), compute the privacy parameter σ by the RDP composition in (Wang et al., 2019) for the $(C+1)$ repeated use of the Gaussian mechanism.

Step 2. Privatize the random feature mean embeddings via $\mathcal{M}_{weights}$ and $\mathcal{M}_{\mathbf{m}_c}$.

Step 3. Train the generator by minimizing eq. 12

As the norm of $\hat{\boldsymbol{\phi}}$ is bounded by 1, the sensitivity of $\mathbf{m}_c$ (eq. 8) is $\Delta_{\mathbf{m}_c} = \frac{2}{m}$.

During the training, we will need to perform $\mathcal{M}_{weights}$ once, and $\mathcal{M}_{\mathbf{m}_c}$ as many times as the number of classes. Hence, we divide our privacy budget into $C+1$ compositions of the Gaussian mechanisms.

Now the objective function to minimize is modified to

$$\widehat{\mathrm{MMD}}^2_{rf}(P^{DP}_{\mathbf{x},\mathbf{y}}, Q_{\tilde{\mathbf{x}}_{\boldsymbol{\theta}},\tilde{\mathbf{y}}_{\boldsymbol{\theta}}}) = \left\| \widetilde{\boldsymbol{\mu}}^{DP}_{P_{\mathbf{x},\mathbf{y}}} - \widehat{\boldsymbol{\mu}}_{Q_{\mathbf{x},\mathbf{y}}} \right\|_2^2, \tag{12}$$

where $\widetilde{\boldsymbol{\mu}}^{DP}_{P_{\mathbf{x},\mathbf{y}}} = \left[\frac{1}{\tilde{\omega}_1}\tilde{\mathbf{m}}_1, \cdots \frac{1}{\tilde{\omega}_C}\tilde{\mathbf{m}}_C \right]$. Our algorithm is summarized in Algorithm 1.

Note that by privatizing the weights and the mean embedding separately, we can get the benefit of sensitivity being on the order of $1/m$, rather than on the order of $1/m_c$ where the latter could hamper the training performance as in highly imbalanced datasets m_c can be very small resulting in a high additive noise variance.

3.2. DP-MERF for heterogeneous data

To handle heterogeneous data consisting of continuous variables denoted by $\mathbf{x}_{con}$ and discrete variables denoted by $\mathbf{x}_{dis}$, we consider the sum of two existing kernels, $k((\mathbf{x}_{con}, \mathbf{x}_{dis}), (\mathbf{x}'_{con}, \mathbf{x}'_{dis})) = k_{con}(\mathbf{x}_{con}, \mathbf{x}'_{con}) + k_{dis}(\mathbf{x}_{dis}, \mathbf{x}'_{dis})$, where k_{con} is a kernel for continuous variables and k_{dis} is a kernel for discrete variables.

As before, we could use the Gaussian kernel for $k_{con}(\mathbf{x}_{con}, \mathbf{x}'_{con}) = \hat{\boldsymbol{\phi}}(\mathbf{x}_{con})^\top \hat{\boldsymbol{\phi}}(\mathbf{x}'_{con})$ and a normalized polynomial kernel with order-1, $k_{dis}(\mathbf{x}_{dis}, \mathbf{x}'_{dis}) = \frac{1}{d_{dis}} \mathbf{x}_{dis}{}^\top \mathbf{x}'_{dis}$ for one-hot-encoded values $\mathbf{x}_{dis}$ and the length of $\mathbf{x}_{dis}$ being d_{dis}. This normalization is to match the importance of the two kernels in the resulting mean embeddings. Under these kernels, we can approximate the mean embeddings using random features

$$\widehat{\boldsymbol{\mu}}_{P_{\mathbf{x}}} = \frac{1}{m} \sum_{i=1}^{m} \hat{\mathbf{h}}(\mathbf{x}^{(i)}_{con}, \mathbf{x}^{(i)}_{dis}), \tag{13}$$

where we define $\hat{\mathbf{h}}(\mathbf{x}^{(i)}_{con}, \mathbf{x}^{(i)}_{dis}) := \begin{bmatrix} \hat{\boldsymbol{\phi}}(\mathbf{x}^{(i)}_{con}) \\ \frac{1}{\sqrt{d_{dis}}} \mathbf{x}^{(i)}_{dis} \end{bmatrix}$ from the definition of kernel (See Appendix for derivation). In summary, for generating input and output pairs jointly when the input features are heterogeneous, we run Algorithm 1 with three changes: (a) redefine $\hat{\mathbf{f}}(\mathbf{x}, \mathbf{y})$ in eq. 6 as $\mathrm{vec}(\hat{\mathbf{h}}(\mathbf{x}_{con}, \mathbf{x}_{dis}) \mathbf{f}(\mathbf{y})^\top)$; (b) redefine $\mathbf{m}_c$ in eq. 11 as $\frac{1}{m}\sum_{i \in c_c}^{m_c} \hat{\mathbf{h}}(\mathbf{x}_i)$; and (c) change the sensitivity of $\mathbf{m}_c$ to $\Delta_{\mathbf{m}_c} = \frac{2\sqrt{2}}{m}$ (see Appendix for proof).

3.3. DP-MERF for image data

Following the convention of the machine learning literature for image data generation, we introduce an encoder which reduces the dimensionality of the high-dimensional image data, denoted by $\mathbf{e}_{\boldsymbol{\tau}} : \mathbf{x} \mapsto \mathbf{g}$, where $\mathbf{x} \in \mathbb{R}^{D_{\mathbf{x}}}$ and $\mathbf{g} \in \mathbb{R}^{D_{\mathbf{g}}}$ and $D_{\mathbf{x}} \gg D_{\mathbf{g}}$. The encoder is parameterized by $\boldsymbol{\tau}$. Simiarly, we impose a decoder that can map the low-dimensional code $\mathbf{g}$ to the data space, $\mathbf{d}_{\boldsymbol{\kappa}} : \mathbf{g} \mapsto \mathbf{x}$ where the decoder is parameterized by $\boldsymbol{\kappa}$. We then introduce a generator that can produce the low-dimensional code which can be transformed to the data space through the decoder. Our method employs two mechanisms below.

$\mathcal{M}_{DP-SGD}$: We train an auto-encoder by minimizing the pixel-wise cross-entropy between the raw pixels and the reconstructed pixels. What's important here is that we employ non-private SGD for the encoder update, while we employ DP-SGD for the decoder update. In our algorithm, we do not need a private encoder as in the mechanism below we will add noise to the embedding by taking into account any one datapoint's contribution to the trained encoder. Hence, we spend less amount of privacy budget compared to algorithms that require perturbing both encoder and decoder.

Using a trained encoder, we now match the random feature mean embeddings on the true codes (codes from the true data) and the generated codes through the generator, $G_{\boldsymbol{\theta}}(\mathbf{z}_{\mathbf{g}_i}, \mathbf{z}_{\mathbf{y}_i}) \mapsto (\mathbf{g}, \mathbf{y})$. The random feature mean embedding is $\widehat{\boldsymbol{\mu}}_{P_{\mathbf{g},\mathbf{y}}} = \frac{1}{m}\sum_{i=1}^m \hat{\mathbf{f}}(\mathbf{e}_{\boldsymbol{\tau}}(\mathbf{x}_i), \mathbf{y}_i)$.

Definition 3.3 ($\mathcal{M}_{\hat{\boldsymbol{\mu}}}$). *The mechanism takes a dataset $\mathcal{D}$ and computes eq. 8. It outputs the privatized quantity given a privacy parameter σ_{gen} and the sensitivity $\Delta_{\widehat{\boldsymbol{\mu}}_{P_{\mathbf{g},\mathbf{y}}}}$,*

$$\widehat{\boldsymbol{\mu}}^{DP}_{P_{\mathbf{g},\mathbf{y}}} = \widehat{\boldsymbol{\mu}}_{P_{\mathbf{g},\mathbf{y}}} + \mathcal{N}(0, \sigma^2_{gen} \Delta^2_{\widehat{\boldsymbol{\mu}}_{P_{\mathbf{g},\mathbf{y}}}} \mathbf{I}). \tag{14}$$

Using the product of two kernels for the joint distribution on the input and output pairs, as before, we arrive at $\hat{\mathbf{f}}(\mathbf{e}_{\boldsymbol{\tau}}(\mathbf{x}_i), \mathbf{y}_i) := \mathrm{vec}(\hat{\boldsymbol{\phi}}(\mathbf{e}_{\boldsymbol{\tau}}(\mathbf{x}_i)) \mathbf{f}(\mathbf{y}_i)^\top)$. As the random features $\hat{\boldsymbol{\phi}}$ are norm bounded (i.e., norm 1), any one datapoint's contribution to the trained encoder $\mathbf{e}$ is also bounded by 1, resulting in $\Delta_{\widehat{\boldsymbol{\mu}}_{P_{\mathbf{g},\mathbf{y}}}} = \frac{2}{m}$ (See Appendix for proof).

Now the objective function to minimize is modified to

$$\widehat{\mathrm{MMD}}^2_{rf}(P^{DP}_{\mathbf{g},\mathbf{y}}, Q_{\tilde{\mathbf{g}}_{\boldsymbol\theta},\tilde{\mathbf{y}}_{\boldsymbol\theta}}) = \left\| \widehat{\boldsymbol\mu}^{DP}_{P_{\mathbf{g},\mathbf{y}}} - \widehat{\boldsymbol\mu}_{Q_{\mathbf{g},\mathbf{y}}} \right\|_2^2, \quad (15)$$

where $\widehat{\boldsymbol\mu}_{Q_{\mathbf{g},\mathbf{y}}} = \frac{1}{n}\sum_{i=1}^n \hat{\mathbf{f}}(G_{\boldsymbol\theta}(\mathbf{z}_{\mathbf{g}_i}, \mathbf{z}_{\mathbf{y}_i}))$. Our algorithm is summarized in Algorithm 2.

Algorithm 2 DP-MERF for generating image data

Require: Dataset $\mathcal{D}$, and a privacy level (ϵ, δ)

Ensure: (ϵ, δ)-DP input images and output labels

Step 1: Given (ϵ, δ), compute the privacy parameters σ_{dec} and σ_{gen} using the RDP composition by (Wang et al., 2019) for the repeated use of the Gaussian mechanism in $\mathcal{M}_1$ and $\mathcal{M}_2$.

Step 2: Train the decoder using $\mathcal{M}_{DP-SGD}$ with σ_{dec}

Step 3: Train the generator using $\mathcal{M}_{\hat{\boldsymbol\mu}}$ with σ_{gen}.

A corollary of the RDP composition theorem in Thm. 2.1 combined with Prop. 1 states that Algorithm 2 is DP.

Corollary 3.1. *If $\mathcal{M}_{DP-SGD}$ with σ_{dec} is $(\alpha, \epsilon_1(\alpha))$-RDP and $\mathcal{M}_{\hat{\boldsymbol\mu}}$ with σ_{gen} is $(\alpha, \epsilon_2(\alpha))$-RDP, then the composition of the two is $(\alpha, \epsilon_1(\alpha) + \epsilon_2(\alpha))$-RDP.*

We convert the RDP level to the DP level by Prop. 1.

4. Related work

There are three categories of relevant work to ours. The first category is the differentially private GAN framework and its variants (Xie et al., 2018; Torkzadehmahani et al., 2019; Frigerio et al., 2019; Yoon et al., 2019). The core technique of most of these algorithms is based on DP-SGD, with an exception that (Yoon et al., 2019) is based on the Private Aggregation of Teacher Ensembles (PATE). Unlike these methods, our method does not involve the difficult task of finding the equilibrium between the generator and the discriminator. Our method is not limited to the binary classification problems as in PATE-GAN (Yoon et al., 2019); nor requires a complicated sensitivity computation as in DP-GAN (Xie et al., 2018). Furthermore, our method can produce input and output pairs jointly for supervised learning problems. DP-CGAN (Torkzadehmahani et al., 2019) also considered this case, while their generator generates only the input features conditioning on the labels. There is no other methods aiming at generating data for supervised learning that we are aware of other than DP-CGAN. Hence, we will compare our method to DP-CGAN in Sec. 5.

The second category is the differentially private auto-encoder framework (Abay et al., 2019; Tantipongpipat et al., 2019; Chen et al., 2018), which reduces the dimensionality of the high-dimensional data into a low-dimensional code space via an auto-encoder training and learns a generator which produces codes. Our method also uses an auto-encoder for image data for dimensionality reduction. However, unlike these methods, we train the generator using the mean embeddings with random features.

Table 1. Tabular datasets. num refers to numerical, cat refers to categorical, and ord refers to ordinal variables

dataset	**# samps**	**# classes**	**type**
isolet	4366	2	homogeneous
covtype	406698	7	10 num, 44 cat
epileptic	11500	2	homogeneous
credit	284807	2	homogeneous
cervical	753	2	11 num, 24 cat
census	199523	2	7 num, 33 cat
adult	22561	2	6 num, 8 cat
intrusion	394021	5	8 cat, 6 ord, 26 num

The third category is the framework of kernel methods with differential privacy. Balog et al. (2018) proposed to use the reduced set method in conjunction with random features for sharing DP mean embeddings, but generative models are not the part of their algorithms. Sarpatwar et al. (2019) also used the random feature representations for the DP distributed data summarization to take into account covariate shifts, but not for the DP data generation.

5. Experiments

The experiments present robustness of the method in producing a diverse range of data both in private and non-private settings. We first train a generator using either DP-MERF or DP-CGAN, and obtain *synthetic* data samples, which we use to train 12 predictive models (see Table). We then use these trained models to predict the labels of *real* test data. As comparison metrics, we use ROC (area under the receiver operating characteristics curve) and ROC (area under the precision recall curve) for binary-labeled data. We use F1 score and prediction accuracy for multiclass-labeled data. As a baseline, we also show the performance of the models trained with the real training data. All the numbers shown in the tables are the average over 5 independent runs.

5.1. Heterogeneous and homogenous tabular data

We begin the experiments with a set of tabular data which contain real-world information. The datasets we consider contain either only numerical data (homogenous) or both numerical and categorical data (including ordinal data such as education), which we call heterogenous datasets. The numerical features which are both discrete and continuous

Table 2. Performance comparison on Intrusion dataset.

	Real	**DP-CGAN** (non-priv)	**DP-CGAN** $(1, 10^{-5})$-DP	**DP-MERF** (non-priv)	**DP-MERF** $(1, 10^{-5})$-DP
Logistic Regression	0.948	0.71	0.567	0.926	0.94
Gaussian Naive Bayes	0.757	0.503	0.215	0.804	0.736
Bernoulli Naive Bayes	0.927	0.693	0.475	0.822	0.755
Linear SVM	0.983	0.639	0.915	0.922	0.937
Decision Tree	0.999	0.496	0.153	0.862	0.952
LDA	0.99	0.224	0.652	0.91	0.95
Adaboost	0.947	0.898	0.398	0.924	0.503
Bagging	1	0.499	0.519	0.914	0.956
Random Forest	1	0.497	0.676	0.941	0.943
GBM	0.999	0.501	0.255	0.924	0.933
MLP	0.997	0.923	0.733	0.933	0.957
XGBoost	0.999	0.886	0.751	0.921	0.933
Average	0.962	0.622	0.526	**0.9**	**0.875**

Table 3. Performance comparison on Tabular datasets.

	Real	**DP-CGAN** (non-priv)	**DP-MERF** (non-priv)	**DP-CGAN** $(1, 10^{-5})$-DP	**DP-MERF** $(1, 10^{-5})$-DP
	ROC/PRC	ROC/PRC	ROC/PRC	ROC/PRC	ROC/PRC
adult	0.73/0.639	0.519/0.451	0.653/0.57	0.509/0.444	0.65/0.564
census	0.747/0.415	0.646/0.2	0.692/0.369	0.655/0.216	0.686/0.358
cervical	0.786/0.493	0.587/0.251	0.896/0.737	0.519/0.2	0.545/0.184
credit	0.923/0.874	0.801/0.432	0.898/0.774	0.664/0.356	0.772/0.637
epileptic	0.797/0.617	0.49/0.19	0.616/0.335		0.611/ 0.34
isolet	0.893/0.728	0.622/0.264	0.733/0.424		0.547/0.404
	F1	F1	F1	F1	F1
covtype	0.643	0.236	0.513	0.285	0.467
intrusion	0.963	0.43	0.856	0.302	0.85

values. The categorical features can have two classes (e.g. whether a person smokes or not) or several classes (e.g. country of origin). The output labels are also categorical; we include datasets with both binary and multiclass labels. Table 1 summarizes the datasets. Table 2 shows the performance of the 12 predictive models trained by the sampled from DP-MERF and DP-CGAN at the level of $(1, 10^{-5})$-DP, compared to the real training and test data. Table 3 shows the average across the 12 predictive models trained by the sampled from DP-MERF and DP-CGAN at the level of $(1, 10^{-5})$-DP. DP-MERF produces high-quality samples which are only a few percentage points short of the real-world data. The method works well both with numerical and categorical data. In the private setting, we perturb the mean embedding of the true data once with the privacy budget $\epsilon = 1$ and $\delta = 10^{-5}$, resulting in a relatively small drop in evaluation metrics.

5.2. High-dimensional image data

[**MP:** Frederik will add descritions here.] Table 4 shows the performance comparison on the MNIST and Fashion-MNIST datasets in terms of the prediction performance. Generated samples under each case are shown in Fig. 1.

Table 4. Performance on image data. Numbers outside parenthesis are classification accuracy; and those inside parenthesis are F1-score.

	Real data	DP-CGAN $\epsilon = 9.6$	DP-MERF+AE $\epsilon = 9.6$	DP-MERF $\epsilon = 9.6$	DP-MERF $\epsilon = 2.9$	DP-MERF $\epsilon = 1.3$
MNIST	0.87 (0.86)	0.50 (0.48)	0.43 (0.41)	**0.58** (**0.57**)	0.55 (0.54)	0.53 (0.52)
FashionMNIST	0.78 (0.77)	0.39 (0.37)	0.44 (0.41)	**0.52** (**0.51**)	0.51 (0.49)	0.48 (0.46)

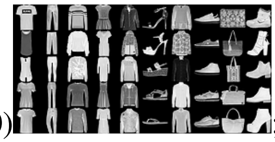
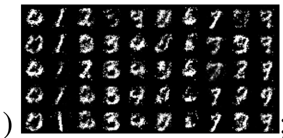
(d0) ; [below=-0.1 of d0] (f0); [right=-0.1 of d0] (d1) ;

Figure 1. Generated samples with $(9.6, 10^{-5})$-DP

6. Summary and Discussion

We proposed a simple and practical algorithm using the random feature representation of kernel mean embeddings for DP data generation. Our method requires a significantly lower privacy budget to produce quality data samples compared to DP-CGAN, tested on 8 tabular data and 2 image data. The metrics we used were targeting at supervised learning tasks. In future work, we plan to test our algorithm in more subtle metrics such as measuring the diversity of generated samples and the ability to cover all the modes of the data distribution.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS 16, pp. 308318, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978318.

Abay, N. C., Zhou, Y., Kantarcioglu, M., Thuraisingham, B., and Sweeney, L. Privacy preserving synthetic data release using deep learning. In Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., and Ifrim, G. (eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 510–526, Cham, 2019. Springer International Publishing. ISBN 978-3-030-10925-7.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan. *ArXiv*, abs/1701.07875, 2017.

Balog, M., Tolstikhin, I., and Schölkopf, B. Differentially private database release via kernel mean embeddings. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 423–431. PMLR, July 2018.

Chen, Q., Xiang, C., Xue, M., Li, B., Borisov, N., Kaafar, D., and Zhu, H. Differentially private data generative models. *CoRR*, abs/1812.02274, 2018.

Csiszr, I. and Shields, P. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):417–528, 2004. ISSN 1567-2190. doi: 10.1561/0100000004.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *Eurocrypt*, volume 4004, pp. 486–503. Springer, 2006.

Frigerio, L., de Oliveira, A. S., Gomez, L., and Duverger, P. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *ICT Systems Security and Privacy Protection - 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings*, pp. 151–164, 2019. doi: 10.1007/978-3-030-22312-0_11.

Gao, H. and Huang, H. Joint generative moment-matching network for learning structural latent code. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 2121–2127. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/293.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. A kernel two-sample test. 13:723–773, 2012.

Hardt, M., Ligett, K., and Mcsherry, F. A simple and practical algorithm for differentially private data release. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 2339–2347. Curran Associates, Inc., 2012.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. Mmd gan: Towards deeper understanding of moment matching network. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS17, pp. 22002210, Red Hook, NY, USA, 2017a. Curran Associates Inc. ISBN 9781510860964.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Poczos, B. Mmd gan: Towards deeper understanding of moment matching network. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2203–2213. Curran Associates, Inc., 2017b.

Mironov, I. Rnyi differential privacy. In *Proceedings of 30th IEEE Computer Security Foundations Symposium (CSF)*, pp. 263–275, 2017. URL https://arxiv.org/abs/1702.07476.

Mohammed, N., Chen, R., Fung, B. C., and Yu, P. S. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pp. 493–501, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020487.

Nowozin, S., Cseke, B., and Tomioka, R. f-gan: Training generative neural samplers using variational divergence minimization. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 271–279, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9.

Papernot, N., Abadi, M., Erlingsson, U., Goodfellow, I., and Talwar, K. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *Proceedings of the International Conference on Learning Representations (ICLR)*, April 2017. URL http://arxiv.org/abs/1610.05755.

Park, M., Foulds, J., Choudhary, K., and Welling, M. DP-EM: Differentially Private Expectation Maximization. In Singh, A. and Zhu, J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 896–904, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.

Park, N., Mohammadi, M., Gorde, K., Jajodia, S., Park, H., and Kim, Y. Data synthesis based on generative adversarial networks. *Proc. VLDB Endow.*, 11(10):10711083, June 2018. ISSN 2150-8097. doi: 10.14778/3231751.3231757.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pp. 1177–1184, 2008.

Rudin, W. *Fourier Analysis on Groups: Interscience Tracts in Pure and Applied Mathematics, No. 12*. Literary Licensing, LLC, 2013.

Sarpatwar, K., Shanmugam, K., Ganapavarapu, V. S., Jagmohan, A., and Vaculin, R. Differentially private distributed data summarization under covariate shift. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 14432–14442. Curran Associates, Inc., 2019.

Sutherland, D. J. and Schneider, J. On the error of random fourier features. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, UAI15, pp. 862871, Arlington, Virginia, USA, 2015. AUAI Press. ISBN 9780996643108.

Szabó, Z. and Sriperumbudur, B. K. Characteristic and universal tensor product kernels. *Journal of Machine Learning Research*, 18(233):1–29, 2018.

Tantipongpipat, U., Waites, C., Boob, D., Siva, A. A., and Cummings, R. Differentially private mixed-type data generation for unsupervised learning, 2019.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

Torkzadehmahani, R., Kairouz, P., and Paten, B. Dp-cgan: Differentially private synthetic data and label generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

Wang, Y.-X., Balle, B., and Kasiviswanathan, S. P. Subsampled renyi differential privacy and analytical moments accountant. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1226–1235. PMLR, 16–18 Apr 2019.

Xiao, Y., Xiong, L., and Yuan, C. Differentially private data release through multidimensional partitioning. In Jonker, W. and Petković, M. (eds.), *Secure Data Management*, pp. 150–168, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15546-8.

Xie, L., Lin, K., Wang, S., Wang, F., and Zhou, J. Differentially private generative adversarial network. *CoRR*, abs/1802.06739, 2018.

Yoon, J., Jordon, J., and van der Schaar, M. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2019.

Zhang, Y.-Y., Shen, C.-M., Feng, H., Fletcher, P. T., and Zhang, G.-X. Generative adversarial networks with joint distribution moment matching. *Journal of the Operations Research Society of China*, 7(4):579–597, Dec 2019. ISSN 2194-6698. doi: 10.1007/s40305-019-00248-x.

Zhu, T., Li, G., Zhou, W., and Yu, P. S. Differentially private data publishing and analysis: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 29(8): 1619–1638, Aug 2017. ISSN 1041-4347. doi: 10.1109/TKDE.2017.2697856.

Supplementatry material for Differentially Private Random Feature Mean Embeddings for Simple & Practical Synthetic Data Generation

7. Derivation of feature maps for a product of two kernels

$$
\begin{aligned}
&k((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) \\
&= k_{\mathbf{x}}(\mathbf{x},\mathbf{x}')k_{\mathbf{y}}(\mathbf{y},\mathbf{y}'), \text{ product of two kernels} \\
&\approx \hat{\boldsymbol{\phi}}(\mathbf{x}')^\top \hat{\boldsymbol{\phi}}(\mathbf{x})\mathbf{f}(\mathbf{y})^\top \mathbf{f}(\mathbf{y}'), \text{ random features for kernel } k_{\mathbf{x}} \\
&= \mathrm{Tr}\left(\hat{\boldsymbol{\phi}}(\mathbf{x}')^\top \hat{\boldsymbol{\phi}}(\mathbf{x})\mathbf{f}(\mathbf{y})^\top \mathbf{f}(\mathbf{y}')\right), \\
&= \mathrm{vec}(\hat{\boldsymbol{\phi}}(\mathbf{x}')\mathbf{f}(\mathbf{y}')^\top)^\top \mathrm{vec}(\hat{\boldsymbol{\phi}}(\mathbf{x})\mathbf{f}(\mathbf{y})^\top) = \hat{\mathbf{f}}(\mathbf{x}',\mathbf{y}')^\top \hat{\mathbf{f}}(\mathbf{x},\mathbf{y})
\end{aligned}
$$

8. Derivation of feature maps for a sum of two kernels

$$
\begin{aligned}
&k((\mathbf{x}_{con},\mathbf{x}_{dis}),(\mathbf{x}'_{con},\mathbf{x}'_{dis})) \\
&= k_{con}(\mathbf{x}_{con},\mathbf{x}'_{con}) + k_{dis}(\mathbf{x}_{dis},\mathbf{x}'_{dis}), \\
&\approx \hat{\boldsymbol{\phi}}(\mathbf{x}_{con})^\top \hat{\boldsymbol{\phi}}(\mathbf{x}'_{con}) + \tfrac{1}{\sqrt{d_{dis}}}\mathbf{x}_{dis}{}^\top \mathbf{x}'_{dis}, \\
&= \begin{bmatrix} \hat{\boldsymbol{\phi}}(\mathbf{x}_{con}) \\ \frac{1}{\sqrt{d_{dis}}}\mathbf{x}_{dis} \end{bmatrix}^T \begin{bmatrix} \hat{\boldsymbol{\phi}}(\mathbf{x}_{con}) \\ \frac{1}{\sqrt{d_{dis}}}\mathbf{x}_{dis} \end{bmatrix} \\
&= \hat{\mathbf{h}}(\mathbf{x}_{con},\mathbf{x}_{dis})^T \hat{\mathbf{h}}(\mathbf{x}_{con},\mathbf{x}_{dis}).
\end{aligned} \tag{16}
$$

9. Sensitivity of $\mathbf{m}_c$ with heterogeneous data

Recall that $\hat{\mathbf{h}}(\mathbf{x}_{con}^{(i)},\mathbf{x}_{dis}^{(i)}) = \begin{bmatrix} \hat{\boldsymbol{\phi}}(\mathbf{x}_{con}^{(i)}) \\ \frac{1}{\sqrt{d_{dis}}}\mathbf{x}_{dis}^{(i)} \end{bmatrix}$ and $\mathbf{m}_c = \frac{1}{m}\sum_{i\in c_c}^{m_c} \hat{\mathbf{h}}(\mathbf{x}_i)$ where $\mathbf{x}_i$ is the concatenation of $\mathbf{x}_{con}^{(i)}$ and $\mathbf{x}_{dis}^{(i)}$, the sensitivity of $\mathbf{m}_c$ is

$$
\begin{aligned}
\Delta_{\mathbf{m}_c} &= \max_{\mathcal{D},\mathcal{D}'} \left\| \tfrac{1}{m}\sum_{i\in c_c}^{m_c} \hat{\mathbf{h}}(\mathbf{x}_i) - \tfrac{1}{m}\sum_{i\in c_c}^{m_c} \hat{\mathbf{h}}(\mathbf{x}'_i) \right\|_2, \\
&= \max_{\mathbf{x}_n,\mathbf{x}'_n} \left\| \tfrac{1}{m}\begin{bmatrix} \hat{\boldsymbol{\phi}}(\mathbf{x}_{con}^{(n)}) \\ \frac{1}{\sqrt{d_{dis}}}\mathbf{x}_{dis}^{(n)} \end{bmatrix} - \tfrac{1}{m}\begin{bmatrix} \hat{\boldsymbol{\phi}}(\mathbf{x}'_{con}{}^{(n)}) \\ \frac{1}{\sqrt{d_{dis}}}\mathbf{x}'_{dis}{}^{(n)} \end{bmatrix} \right\|_2, \\
&\leq \max_{\mathbf{x}_n} \tfrac{2}{m} \left\| \begin{bmatrix} \hat{\boldsymbol{\phi}}(\mathbf{x}_{con}^{(n)}) \\ \frac{1}{\sqrt{d_{dis}}}\mathbf{x}_{dis}^{(n)} \end{bmatrix} \right\|_2, \\
&\leq \max_{\mathbf{x}_{dis}^{(n)}} \tfrac{2}{m} \sqrt{1 + \tfrac{1}{d_{dis}}\sum_{j=1}^{d_{dis}} (\mathbf{x}_{dis,j}^{(n)})^2}, \text{ since} \|\hat{\boldsymbol{\phi}}(\cdot)\|_2 = 1 \\
&\leq \frac{2\sqrt{2}}{m},
\end{aligned} \tag{17}
$$

where the list line is because $\mathbf{x}_{dis}$ is a vector of binary variables.

10. Sensitivity of $\widehat{\boldsymbol{\mu}}_{P_{\mathbf{g},\mathbf{y}}}$ for image data

Using the product of two kernels

$$
\begin{aligned}
&\Delta_{\widehat{\boldsymbol{\mu}}_{P_{\mathbf{g},\mathbf{y}}}} \\
&= \max_{\mathcal{D},\mathcal{D}'} \left\| \tfrac{1}{m}\sum_{i=1}^{m} \hat{\mathbf{f}}(\mathbf{e}_{\boldsymbol{\tau}}(\mathbf{x}_i),\mathbf{y}_i) - \tfrac{1}{m}\sum_{i=1}^{m} \hat{\mathbf{f}}(\mathbf{e}_{\boldsymbol{\tau}}(\mathbf{x}'_i),\mathbf{y}'_i) \right\|_2, \\
&= \max_{\mathbf{x}_n,\mathbf{x}'_n} \left\| \begin{bmatrix} \mathbf{0} & \cdots \tfrac{1}{m}\hat{\boldsymbol{\phi}}(\mathbf{e}_{\boldsymbol{\tau}}(\mathbf{x}_n)) & \cdots & \tfrac{1}{m}\hat{\boldsymbol{\phi}}(\mathbf{e}_{\boldsymbol{\tau}}(\mathbf{x}'_n)) \cdots \mathbf{0} \end{bmatrix} \right\|
\end{aligned}
$$

where only two columns are non-zero, as there are only two datapoints difference in two datasets if the labels of these two points are different. As the random features are norm bounded (by 1), the sensitivity is $\frac{\sqrt{2}}{m}$. If the labels of those two points are the same, only one column is non-zero, where the value is $\frac{1}{m}\hat{\boldsymbol{\phi}}(\mathbf{e}_{\boldsymbol{\tau}}(\mathbf{x}_n)) - \frac{1}{m}\hat{\boldsymbol{\phi}}(\mathbf{e}_{\boldsymbol{\tau}}(\mathbf{x}'_n))$. Hence, the sensitivity is $\frac{2}{m}$. Therefore the worse case upper bound is $\Delta_{\widehat{\boldsymbol{\mu}}_{P_{\mathbf{g},\mathbf{y}}}} = \frac{2}{m}$.

11. Rényi differential privacy

Definition 11.1 (α-Rényi Divergence)**.** *For two probability distributions P, Q that have the same support, the α Rényi*

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country [2]**AUTHORERR: Missing \icmlaffiliation.** . Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

divergence is

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} \log_{x \sim Q(x)} \left(\frac{P(x)}{Q(x)} \right)^\alpha \qquad (18)$$

for $\alpha \in (1, \infty)$.

12. Description of the datasets

12.1. Covtest

****************** covtype

The number of attributes is 55, out of which 10 are numerical 45 are categorical. The number of samples is 581012. Notes: n$_featuresgoodbetween100and5000, withbathc0.01andtrained10000.563,' how_many_epochs_arg'$: $1000,' mini_batch_arg'$: $0.05,' n_features_arg'$: 500 $* * * * * * * * * * * * * * * * * *epilepticROConrealtestdatais0.5569809947080179PRConrealtestdatais0.2860654400815256$

'how$_many_epochs_arg'$: $2000,' mini_batch_arg'$: $500,' n_features_arg'$: $500avbest$

'how$_many_epochs_arg'$: $1000,' mini_batch_arg'$: $1000,' n_features_arg'$: $50ROCis0.6571148022863209PRCis0.3312778660464996singbest$

************** credit card fraud (284807, 29) ROC on real test data is 0.9365079365079365 PRC on real test data is 0.8835421888053467

best single 'how$_many_epochs_arg'$: $2000,' mini_batch_arg'$: $0.5,' n_features_arg'$: $500ROCis0.9285714285714286PRCis0.868984962406015* * * * * * * * * *census(199523, 40)features$

********************* cervical raw input features (753, 34)