The VLDB Journal (2018) 27:201–223
https://doi.org/10.1007/s00778-017-0492-3

REGULAR PAPER

PrivPfC: differentially private data publication for classification

Dong Su[1] · Jianneng Cao[2] · Ninghui Li[1] · Min Lyu[3]

Received: 25 September 2016 / Revised: 19 November 2017 / Accepted: 14 December 2017 / Published online: 2 February 2018

Abstract
In this paper, we tackle the problem of constructing a differentially private synopsis for the classification analysis. Several state-of-the-art methods follow the structure of existing classification algorithms and are all iterative, which is suboptimal due to the locally optimal choices and division of the privacy budget among many sequentially composed steps. We propose PrivPfC, a new differentially private method for releasing data for classification. The key idea underlying PrivPfC is to privately select, in a single step, a grid, which partitions the data domain into a number of cells. This selection is done by using the exponential mechanism with a novel quality function, which maximizes the expected number of correctly classified records by a histogram classifier. PrivPfC supports both the binary classification and the multiclass classification. Through extensive experiments on real datasets, we demonstrate PrivPfC's superiority over the state-of-the-art methods.

Keywords Differential privacy · Classification · Privacy preserving data publishing

1 Introduction

Classification is an important tool for data analysis. However, publishing parameters of a classifier learned from a dataset can result in privacy concerns [12,13]. One way to deal with the privacy concerns is to conduct classification while satisfying differential privacy [11]. Differentially private algorithms work by injecting randomness into the computation of summary statistics which are computed from the underlying sensitive data, which makes the distribution of the noisy results being relatively insensitive to any single record change in the original dataset. This ensures that the adversary cannot infer the membership information of any particular record with high confidence (controlled by parameter ϵ), even if he/she has complete knowledge of all other records of the dataset [11,30]. On the other hand, the noisy results should be close to the unperturbed ones in order to be useful in practice. Therefore, the goal of a differentially private data classification mechanism is to maximize classification accuracy, while satisfying the privacy guarantees. Several approaches for learning classifiers while satisfying differential privacy have been proposed in recent years. Some methods compute a classifier as the output [4,6,7,14,19,37]. Other methods [26,34,38] publish a synopsis of the dataset, often in the form of a noisy histogram, so that synthetic datasets can be generated and classifiers can be learned from these synthetic datasets. Publishing a synopsis enables additional exploratory and predictive data analysis tasks to be performed and can be argued to be more preferred.

Publishing noisy histograms for one-dimensional or two-dimensional datasets has been studied extensively in recent years, see [17] for a recent survey. However, as observed in [28,34], these approaches do not work well when the number of attributes/dimensions goes above a few. Many datasets that are of interest have multiple attributes. For a multi-attribute dataset with more than a dozen or so attributes, publishing a histogram with all the attributes results in a sparse histogram where noises may overwhelm the true counts. Therefore, it is necessary to select a subset of the attributes that are "useful" for the intended data analysis tasks, and to determine how to discretize the attributes. These selections partition the domain into a number of cells. We call the result a "grid."

✉ Dong Su
su17@cs.purdue.edu

Jianneng Cao
caojn@i2r.a-star.edu.sg

Ninghui Li
ninghui@cs.purdue.edu

Min Lyu
lvmin05@ustc.edu.cn

1 Purdue University, West Lafayette, IN, USA

2 Institute for Infocomm Research, Singapore, Singapore

3 University of Science and Technology of China, Hefei, China

Once a grid is selected, the next step is straightforward: one adds Laplace noises [11] to the cell counts to produce a noisy histogram.

Thus, the key design choice in algorithms for publishing noisy histograms is how to select a suitable grid. Publishing a histogram is similar to performing generalization for the purpose of achieving k-anonymization, since the exact attribute values for records within a cell no longer matter, and only the cell boundary and the number of records in a cell matter. A key challenge studied in research on k-anonymization was also how to find a high-quality grid [2,21,22]. However, these proposed methods for k-anonymization have been found to be vulnerable to attacks exploiting background information, e.g., the minimality attack [8,35]. Fortunately, an approach to select a grid while satisfying differential privacy, as proposed in this paper, can help to defend against these attacks [10,11].

In this paper we propose the Private Publication for Classification (PrivPfC), a differentially private method for publishing projected histograms for classification. On the key decision of how to select a grid, PrivPfC differs from previous approaches in that it selects a high-quality grid in a single step, whereas previous approaches use an iterative process and as a consequence suffer from two weaknesses. First, an iterative process has to divide the privacy budget among all the iterations, causing the choice made in each iteration to have significant noise. Second, an iterative process is a greedy process and tends to result in a suboptimal global choice even without considering noises.

The exponential mechanism [24] enables the private selection of a grid in a single step. However, there are a number of challenges to use it effectively. One needs to generate a pool of candidate grids that include the high-quality grids, without making the candidate pool too large, which affects both running time and accuracy. Furthermore, one needs a quality function that can effectively identify high-quality grids and simultaneously has a low sensitivity.

From the perspective of classification which maps the set of feature attributes into the class attribute, PrivPfC can be seen as projecting the full set of feature attributes onto a small number of selected informative attributes that are highly correlated with the response attribute. Since for relational data the number of informative attributes is always not large, PrivPfC can identify most of them and construct histogram to summarize data distribution with high utility for classification. Since PrivPfC does not rely on any existing classification algorithm to work, the data published by our PrivPfC method can be used for performing all existing classification tasks.

The first contribution of this paper is PrivPfC, an algorithm for privately publishing noisy histograms optimized for classification. PrivPfC has two novel ideas. One is a method to enumerate through candidate grids when given a cap on how many grids the algorithm is allowed to consider. And the other one is a new quality function that enables the selection of a high-quality “grid”. This quality function considers the impact of injected noises on the classification accuracy, adapts to the privacy parameter ϵ and has a low sensitivity.

Our second contribution is that, through extensive experiments on real datasets, we have compared PrivPfC against other state-of-the-art methods for data publishing as well as private classification, demonstrating that PrivPfC improves the state-of-the-art. We also analyze variants of competing algorithms, showing that their weaknesses come from the iterative structure of their algorithms. We note that the fact that PrivPfC outperforms state-of-the-art algorithms specifically designed for privately publishing classifiers is quite counter-intuitive. PrivPfC publishes a histogram, which contains more information than a classifier; thus, one would expect the classifiers it produces are less accurate. Experimental results demonstrate otherwise. We believe this points to the possibility of designing better private classification algorithms by using as few steps as possible, avoiding spreading the privacy budget too thin.

The rest of the paper is organized as follows. In Sect. 2, we give preliminary information about differential privacy. Related works are summarized in Sect. 3. Our PrivPfC approach is presented in Sect. 4. The experimental results are shown in Sect. 5. Section 6 concludes our work.

2 Background

Definition 1 (*ϵ-differential privacy* [10,11]) A randomized mechanism $\mathcal{A}$ gives ϵ-differential privacy, if for any pair of neighboring datasets D and D' and any $S \in Range(\mathcal{A})$,

$$\Pr\left[\mathcal{A}(D) = S\right] \le e^{\epsilon} \cdot \Pr\left[\mathcal{A}(D') = S\right].$$

In this paper we consider 2 datasets D and D' to be neighbors if and only if either $D = D' + t$ or $D' = D + t$, where $D + t$ denotes the dataset resulted from adding the tuple t to the dataset D. We use $D \simeq D'$ to denote this.

There are several primitives for satisfying ϵ-differential privacy. In this paper we use two of them, Laplace mechanism [11] and exponential mechanism [24].

Given an analysis task f which tasks the dataset as the input and outputs a numerical value, the Laplace mechanism perturbs f's output with random noise drawn from the Laplace distribution with the scale parameter proportional to Δ_f, the *global sensitivity* of the function f. Specifically, we use Laplace mechanism $\mathcal{A}_f$ to compute f on a dataset D in a differentially private way, where:

$$\mathcal{A}_f(D) = f(D) + \mathsf{Lap}\left(\frac{\Delta_f}{\epsilon}\right),$$

where $\Delta_f = \max_{D \simeq D'} |f(D) - f(D')|$, and $\Pr[\mathsf{Lap}(\beta) = x] = \frac{1}{2\beta} e^{-|x|/\beta}$.

In the above, $\mathsf{Lap}(\beta)$ denotes a random variable sampled from the zero mean Laplace distribution with scale parameter β.

While the Laplace mechanism provides a solution to handle tasks with numerical outputs, it cannot be applied to those with non-numeric value outputs. This motivates the development of the exponential mechanism [24], which can be applied whether a function's output is numerical or categorical. The exponential mechanism releases a differentially private version of the task f, by sampling from f's output domain O. The sampling probability for each output $o \in O$ is determined based on a user-specified *quality* function $q : \mathcal{D} \times O \to \mathbb{R}$ that assigns a real valued score to one output $o \in O$ when the input dataset is D, where higher scores indicate more desirable outputs. Given the quality function q, its global sensitivity Δ_q is defined as:

$$\Delta_q = \max_{o} \max_{D \simeq D'} |q(D, o) - q(D', o)|.$$

The following method satisfies ϵ-differential privacy:

$$\Pr[r \text{ is selected}] \propto e^{\left(\frac{\epsilon}{2\Delta_q} q(D,o)\right)}.$$

Differential privacy is sequentially composable in the sense that combining multiple mechanisms $\mathcal{A}_1, \ldots, \mathcal{A}_m$ that satisfy differential privacy for $\epsilon_1, \ldots, \epsilon_m$ results in a mechanism that satisfies ϵ-differential privacy for $\epsilon = \sum_i \epsilon_i$. When a task involves multiple steps, each step uses a portion of ϵ so that the sum of these portions is no more than ϵ. Differential privacy is also parallel composable. Suppose the data are partitioned into m arbitrary disjoint subsets, and mechanism $\mathcal{A}_i$ is executed on the i'th subset. The parallel execution of $\mathcal{A}_1, \ldots, \mathcal{A}_m$ satisfies $\max_i(\epsilon_i)$-differential privacy.

3 Related work

Classification is typically formulated as an optimization problem. We use D to denote the input dataset, ω to denote the model parameter, ω^* to denote the desired model parameter, and $J(D, \omega)$ to denote the objective function to be minimized. That is, we want to output $\omega^* = \mathrm{argmin}_\omega J(D, \omega)$. We now discuss techniques that have been developed to perform such optimization tasks while satisfying DP.

3.1 Output perturbation

One natural method is to directly perturb the output of the optimization problem. This requires analyzing the sensitivity of the optimization problem; that is, how much ω^* changes when the input dataset D changes by one tuple. Unfortunately, the sensitivities of the optimization problems for classification tasks such as logistic regression and SVM tend to be so high that such output perturbation destroys utility [6,7].

3.2 Objective perturbation

An interesting approach, first introduced in [6], is to perturb the optimization objective function so that solving it results in a private solution. We now discuss two such techniques.
Adding a Linear Noise Term to the Optimization Objective Function. Chaudhuri et al. [6,7] proposed to generate a vector of Gamma noises and add the inner product of this vector and the model parameter ω to the optimization objective function. Specifically, the original objective function is,

$$J(D, \omega) = \left(\frac{1}{|D|} \sum_{i=1}^{|D|} L_\omega(x_i) \right) + \lambda c(\omega),$$

where D is the dataset, x_i is the i'th point in D, $|D|$ is the size of D, ω is the model parameter, $L_\omega(x_i)$ defines the loss of the model with parameter ω on the data point x_i, $c(\omega)$ is the regularizer to prevent over-fitting and λ is the regularization parameter to control the degree of regularization. Assuming that both $L_\omega(x_i)$ and $c(\omega)$ are strictly convex and everywhere differentiable for ω, then the perturbed objective function is

$$J^*(D, \omega) = J(D, \omega) + \frac{b^T \omega}{|D|},$$

where b is a random vector sampled from the Gamma distribution with shape parameter d and rate parameter $\frac{2}{|D|\lambda\epsilon}$. Chaudhuri et al. [6,7] showed that solving $\mathrm{argmin}_\omega J^*(D, \omega)$ satisfies ϵ-DP for both logistic regression and SVM.
The functional mechanism Zhang et al. [39] proposed to perturb the objective function by first approximating it using a polynomial, and then perturbing every coefficient of the polynomial. The number of coefficients depends on both the number of attributes and the degree of the polynomial.

When applied to linear regression, the scale of Laplace noise is $2(1 + 2d + d^2)$, where d is the number of dimensions of the dataset D. Note that this sensitivity becomes very large as d increases. Adding noises with this magnitude to every coefficient, and then optimizing for that objective function result in poor performances. For other regression tasks, e.g., logistic regression, where the objective function is not a polynomial with finite order, Zhang et al. [39] proposed to use the first two terms of Taylor expansion to approximate this kind of objective function.

3.3 Make an existing algorithm private

Another approach for differentially private optimization is to take a non-private optimization algorithm, and to apply the Laplace mechanism or the exponential mechanism to ensure that every step is private. Often times, one takes an iterative algorithm for an optimization task, and then makes each iteration private.

DiffPID3: differential private ID3 algorithm for decision tree classification In [4], the algorithm for constructing an ID3 decision tree classifier is made differentially private. When the ID3 algorithm needs to get the number of records with a specific feature value, it queries the SuLQ interface to get the corresponding noise count. The DiffPID3 algorithm in [14] improved this approach by redesigning the classic ID3 classifier construction algorithm to use an attribute quality function that has low sensitivity. Specifically, the DiffPID3 algorithm starts with the most general partition of the underlying dataset. Then, the algorithm chooses the attribute that maximizes the purity metrics (e.g., information gain or Gini index) by using the exponential mechanism. Next, the algorithm splits the dataset with the selected attribute. The same process is applied recursively on each subset of the dataset until there is no further split that improve the purity. In [14], DiffPC-4.5 is proposed as the extension of DiffPID3 to support continuous attributes and pruning.

3.4 Iterative local search via the exponential mechanism

The third approach is to iteratively apply the exponential mechanism to conduct a local search for the optimal model parameter ω^*. In order to do this, one has to generate a candidate set, e.g., by generating multiple perturbations of the current model parameter ω, and then select among the set in a private fashion.

PrivGene: differentially private model fitting using genetic algorithms PrivGene [38] is a general-purpose differentially private model fitting framework based on genetic algorithms. It initializes a candidate set of possible parameters ω and iteratively refines them by mimicking the process of natural evolution. Specifically, in each iteration, PrivGene selects m' parameters from the candidate set and generates from them offsprings by crossover and mutation. The selection is done by using the exponential mechanism with the fitting function of the target model as the quality function. The fitting function measures how well the model with the parameter fits the dataset. Then, it creates a new parameter set, which includes all the offsprings. At the last iteration, a single parameter is selected and outputted as the final result. PrivGene is applied to logistic regression, SVM, and k-means clustering.

PrivLocal: iterative local search A more effective local search algorithm was developed using ideas from the PrivGene paper [38], but does not use features of genetic programming. The algorithm is implemented in the code accompanying the PrivGene paper [38], even though the algorithm did not appear in the paper. Since this algorithm has not appeared in any publication so far, we present it in Algorithm 1.

Algorithm 1 PrivLocal

INPUT D: Dataset, d: number of dimensions of D, J: objective function, ϵ: privacy parameter, r: number of iterations, ω_0: initial parameter, s: perturbation increment (default value: 0.5), $\beta < 1$: scaling parameter (default value: 0.95)
Output ω: selected parameter

```
$\omega \leftarrow \omega_0$
$\epsilon' \leftarrow \epsilon/r$                        ▷ Privacy budget for each iteration
for $i = 1$ to $r$ do
    $\Omega \leftarrow \{\}$                           ▷ Initialize candidate pool
    for $j \in \{1..d\}$ do
        $\omega^1 \leftarrow \omega$ with $j$'s attribute $+ s$
        $\omega^2 \leftarrow \omega$ with $j$'s attribute $- s$
        $\Omega \leftarrow \Omega \cup \{\omega^1, \omega^2\}$
    for $j = 1 \rightarrow |\Omega|$ do
        $q \leftarrow J(D, \Omega_j)$ ▷ Use value of objective function as the quality
        $p_j \leftarrow e^{\frac{q\epsilon'}{2\times\Delta_J}}$      ▷ $\Delta_J$ is the global sensitivity of the objective function $J$
    $\omega \leftarrow$ sample $\Omega_j$ with prob $\frac{p_j}{\sum_j p_j}$
    $s \leftarrow s \cdot \beta$
return $\omega$
```

This algorithm has several interesting ideas. Each round, it uses the exponential mechanism to select a parameter resulted from a single local perturbation that improves the current solution. Compared with PrivGene, which selects multiple candidates (for the purpose of using crossovers to generate the pool of candidates), this means that more privacy budget can be used in each selection. Since only one candidate is selected, there is no crossover. The mutation step takes the form of perturbing the coefficient in a single dimension. That is, each iteration can be viewed as moving along one dimension toward a potentially better parameter. Finally, the perturbation increment s exponentially decays so that the amount of changes decreases. This makes sense as when one starts to converge to the optimal parameter, smaller adjustments are needed.

3.5 Histograms optimized for optimization

Approaches that are most closely related to our proposed method publish a synopsis of the dataset in the form of a noisy histogram, so that synthetic datasets can be generated and optimizers can be learned from these synthetic datasets. Publishing a synopsis enables additional exploratory and predictive data analysis tasks to be performed, and can be argued to be more preferred.

Low-dimensional datasets with numerical attributes Lei [23] proposed a scheme to partition the data space into M equal-width grid cells and then publish noisy counts in each cell. It suggests to choose the number of cells to be $M = \left(\frac{N}{\sqrt{\log N}}\right)^{\frac{2d}{2+d}}$, where N is the dataset size and d is the number of dimensions. This approach does not depend on the privacy parameter ϵ. Qardaji et al. [27] proposed Uniform Griding (UG) for two-dimensional datasets with numerical attributes. UG partitions the space into $M = \frac{N\epsilon}{10}$ of cells. Su et al. [31,32] extended UG to higher-dimensional case, and set $M = \left(\frac{N\epsilon}{10}\right)^{\frac{2d}{2+d}}$.

DiffGen: differentially private anonymization based on generalization Mohammed et al. [26] proposed DiffGen to publish histograms for classification under differential privacy. It consists of 2 steps, partition and perturbation. Given a dataset D and taxonomy trees for each predictor attribute, the partition step starts by generalizing all attributes' values into the topmost nodes in their taxonomy trees. It then iteratively selects one attribute's taxonomy tree node at a time for specialization by using the exponential mechanism. The quality of each candidate specialization is based on the heuristics used by the decision tree constructions, such as information gain and majority class. The partition step terminates after a given number of specializations. The perturbation step injects Laplace noise into each cell of the partition and outputs all the cells with their noisy counts as the noisy synopsis of the data. Privacy budget needs to be evenly distributed to all the specialization steps.

PPH: private projected histogram Vinterbo [34] proposed another data publishing algorithm, called Private Projected Histogram (PPH). PPH first decides how many attributes are to be selected, then incrementally selects attributes via the exponential mechanism to maximize the discernibility of the selected attributes. For each categorical attribute, the full domain is used. For numerical attribute, it uses the formula proposed in [23] to decide how many bins to discretize them. In this method, the number of attributes and how attributes are partitioned are independent of the privacy budget.

PrivBayes: private data release via Bayesian networks Zhang et al. [37] proposed PrivBayes, which publishes a noisy Bayesian network that approximates the data distribution by several low-dimensional marginals (histograms). PrivBayes determines the structure of a Bayesian network by first randomly selecting an attribute as the first node, and then iteratively selecting another attribute to create a new node with up to k nodes already created as the new node's parent nodes. After the structure is determined, PrivBayes perturbs the marginals needed for computing the conditional distributions of the data.

4 PrivPfC Framework

In this section, we present PrivPfC, an algorithm for privately publishing noisy histograms optimized for classification. The main idea of PrivPfC is simple: it uses a single step to privately select a grid which partitions the data domain into a number of cells. It then injects noises to the histogram defined by the selected grid and finally releases the noisy histogram. The key point underlying PrivPfC is how to design a quality function which is used by the exponential mechanism to select a good grid. The quality should satisfy the following design goals: (1) quality score should measure the closeness of a classifier learned from a noisy histogram generated using the grid and a classifier learned from the original data; (2) it should consider the impact of injected noises on the classification accuracy; (3) it should support both binary classification and multiclass classification; (4) it should not rely on any particular classification algorithm. We propose a novel quality function whose quality score is defined by the expected number of correctly classified records by a noisy histogram classifier. And we show that this quality function meets the above design goals.

We consider a dataset with a set of predictor variables and one response variable. The predictor variables can be numerical or categorical. Following [2,15,18,26], for each predictor variable A_i, we assume the existence of a *taxonomy hierarchy* (also called a *generalization hierarchy* in the literature). Fig. 1 shows the taxonomy hierarchies for *Relationship*, a categorical variable and *Education-num*, a numerical variable. In a hierarchy, the root node represents the whole domain of the variable, and a parent node is a generalization (or a cover) of its children. Child nodes under the same parent node are semantically closer to each other than to nodes under a different parent node.

Each *level* of a predictor variable's taxonomy hierarchy forms a partition of its domain. On the basis of the taxonomy hierarchy and its levels, we introduce the notion of a grid.

Definition 2 (*Grid*) Let $\mathsf{A} = \{A_1, \ldots, A_d\}$ be the set of predictor variables in a dataset and $\mathsf{T} = \{T_1, \ldots, T_d\}$ be their taxonomy hierarchies, respectively. Then, a *grid* g is defined by the level used in each predictor variable. Specifically, $g = \langle \ell_1, \ldots, \ell_d \rangle$, where ℓ_i is the index of used level in T_i, $1 \leq \ell_i \leq h_i$ and h_i be the height of T_i, $1 \leq i \leq d$. Such a grid g defines a partitioning of the data domain into *cells* where each predictor variable A_i is partitioned into the values at level ℓ_i in the hierarchy T_i. The set of the cells in the grid g is denoted by $\mathcal{C}_{g,\mathsf{T}}$. In the following, we will abuse notation and

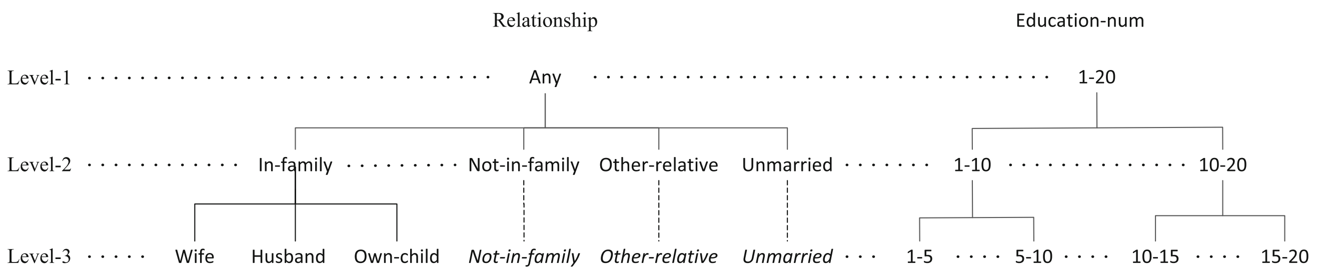

Fig. 1 Taxonomy hierarchies of Relationship attribute and Education-num attribute

write $\mathcal{C}_g$ as shorthand for the set of the cells in g. The *size* of g is the number of cells in $\mathcal{C}_g$, $\mathsf{size}(g) = |\mathcal{C}_g| = \Pi_{i=1}^{d} |T_i[l_i]|$, where $|T_i[l_i]|$ is the number of nodes in the level l_i of the hierarchy T_i. And the number of all possible grids is $\Pi_{i=1}^{d} h_i$.

Example 1 For two taxonomy hierarchies of the Relationship attribute and Education-num attribute (see Fig. 1), we can construct a grid $g = \langle 2, 3 \rangle$, which means that the grid g is defined by the 2nd level of the Relationship hierarchy and the 3rd level of the Education-num hierarchy. The full representation of the grid is {In-family, Not-in-Family, Other-relative, Unmarried} $\times$ {1–5, 6–10, 11–15, 16–20}. And the size of g is $\mathsf{size}(g) = 4 \times 4 = 16$. The number of all possible grids is $3 \times 3 = 9$ since the heights of two taxonomy hierarchies are both 3.

Definition 3 (*Histogram*) Given a dataset D and a grid g, a histogram $\mathsf{H}(D, g)$ partitions D into cells according to g, and in each cell outputs the number of records for each value of the response variable.

PrivPfC publishes $\tilde{\mathsf{H}}(D, g)$, a noisy histogram of the input dataset D, which adds Laplace noise into the counts in the histogram $\mathsf{H}(D, g)$. The key challenge lies in selecting a suitable grid g. Our approach is to define a quality score for each grid, which measures the usefulness for classification of each grid, and apply the exponential mechanism to privately select a grid.

4.1 The quality function

The quality function needs to satisfy the following conditions. First, it needs to accurately measure the desirability of using a particular grid g. Second, it should have a low sensitivity. Intuitively, we want to ensure that classifiers learned from $\tilde{\mathsf{H}}(D, g)$, a noisy histogram of D using g to partition the data domain, are close to classifiers learned from D directly. Furthermore, we desire this to hold regardless of which particular classification algorithm is used and to hold for both the binary classification and the multiclass classification.

We propose to define the quality function to maximize the number of records in D that are classified correctly by the following classifier: for each cell in the grid defined by g, it predicts the class with the highest count according to the noisy histogram $\tilde{\mathsf{H}}(D, g)$. This classifier is in the same spirit as histogram classifiers [9], and we use $HC^{\tilde{\mathsf{H}}(D,g)}$ to denote it. When a grid g is fixed, the noisy histogram includes random noises; therefore, the number of correctly classified records is a random variable. We use the expected value of this random variable as the quality function. Therefore, the quality of the grid g can be defined as:

Definition 4 (*Grid quality*) Given a dataset D with k different class labels, $\mathbb{L} = \{1, 2, \ldots, k\}$, a grid g and ϵ for the parameter of adding Laplace noise to the counts, the grid quality is measured by the expected number of correctly classified records of the histogram classifier $HC^{\tilde{\mathsf{H}}(D,g)}$:

$$\mathsf{gq}(D, g) = \sum_{c \in \mathcal{C}_g} \sum_{i=1}^{k} n_c^i \cdot p_c^i, \tag{1}$$

where $i \in \mathbb{L}$ ranges over the class labels, n_c^i is the number of data points in the cell c with class label i, and p_c^i is the probability that class i is the dominant class in cell c (i.e., with the highest noise count) after injecting Laplace noises. The probability p_c^i is given below:

$$\begin{aligned} p_c^i &= \mathsf{Pr}\left[\text{Class } i \text{ is the dominant class after adding noise}\right] \\ &= \mathsf{Pr}\left[n_c^i + Z_i \geq \max_{j \in \mathbb{L}/\{i\}} \left(n_c^j + Z_j\right)\right] \\ &= \int_{-\infty}^{\infty} \left(\mathsf{Pr}\left[n_c^i + Z_i = x\right] \prod_{j \in \mathbb{L}/\{i\}} \mathsf{Pr}\left[n_c^j + Z_j < x\right] \right) \mathrm{d}x \\ &= \int_{-\infty}^{\infty} \left(f\left(x - n_c^i\right) \prod_{j \in \mathbb{L}/\{i\}} F\left(x - n_c^j\right) \right) \mathrm{d}x, \end{aligned} \tag{2}$$

where Z_i is the Laplace noise added to class i's count, and $f(\cdot)$ and $F(\cdot)$ are, respectively, the probabilistic density function and the cumulative distribution function of the Laplace distribution $\mathrm{Lap}(1/\epsilon)$. The probability p_c^i depends on ϵ, the privacy budget for perturbation, and on the counts of every classes in the cell c.

Intuitively, since the grid quality function gq (Eq. 1) counts number of records, it should have a low sensitivity, since adding or removing a record affects only one of the counts, and changes the count by just 1. However, changing the counts also affects the probabilities. Thus, analyzing the sensitivity of the quality function is quite non-trivial.

4.2 Sensitivity in the binary classification case

We first study the sensitivity of the grid quality (Eq. 1) in the special case where the response variable is binary.

Lemma 1 (Grid quality for binary classification) *Given a dataset D with class labels $\mathbb{L} = \{1, 2\}$, the global sensitivity of the quality function* gq *is bounded by* 1.1.

More specifically, for each ϵ value, the sensitivity is given by

$$\Delta_{gq}(\epsilon) = \max_{x \in \{1,2,3,\ldots\}} f_\epsilon(x), \tag{3}$$

where

$$f_\epsilon(x) = \left|(x-1)\cdot\left(\frac{e^{-\epsilon(x-1)}}{2}\left(1+\frac{\epsilon(x-1)}{2}\right) - \frac{e^{-\epsilon x}}{2}\left(1+\frac{\epsilon x}{2}\right)\right) + \left(1-\frac{e^{-\epsilon x}}{2}\left(1+\frac{\epsilon x}{2}\right)\right)\right|, \tag{4}$$

The global maximum points for $f_\epsilon(x)$, where x ranges over all positive real number, are given below.

$$x^* = \frac{\epsilon e^\epsilon + \sqrt{2-(4-2e^\epsilon)\,e^\epsilon + \epsilon^2 e^\epsilon}}{-\epsilon + \epsilon e^\epsilon}.$$

The proof is deferred to the "Appendix" section. Lemma 1 enables us to compute the sensitivity of the quality function for each ϵ value used for adding noises. Figure 2 shows the calculated sensitivity for 700 different ϵ values in the range of 0.00001 to 100. We note that each time one invokes PrivPfC, the ϵ value is fixed and one can thus compute the sensitivity to be used in the exponential mechanism. Using this instead of 1.1 slightly improves the utility, while satisfying differential privacy.

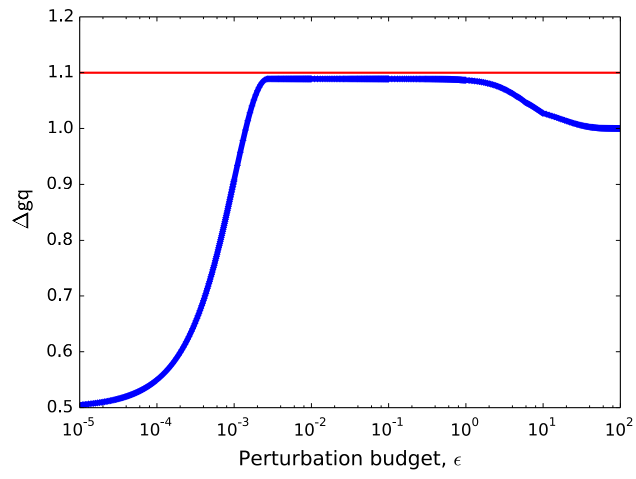

Fig. 2 Illustration of the sensitivity of grid quality (Eq. 3)

4.3 The sensitivity of grid quality in the multiclass classification case

For the general multiclass classification case, where there are more than 2 class labels, deriving an analytical formula similar to Lemma 1 is challenging. Recall that the noisy histogram classifier determines the class label of each cell by ranking all classes according to their noisy counts. The grid quality (Eq. 1) models the process by computing the probability that each class is ranked first after adding noises to each cell. However, since k independent Laplace random variables are involved in this ranking process, getting the closed form of the density of the joint distribution is very challenging. To get a function for the multiclass case whose global sensitivity is easy to bound, we propose a simple and effective approximation of the grid quality (Eq. 5) which for each cell considers only the two classes with the highest counts in that cell.

Definition 5 (*Approximation of grid quality*) Given a dataset D with class labels $\mathbb{L} = \{1, 2, \ldots, k\}$, where $k > 2$, a grid g and ϵ for the parameter of adding Laplacian noises to the counts, the grid quality is

$$\mathsf{gq}(D, g) = \sum_{c \in C_g} n_c^{(1)} \cdot p_c^{(1)} + n_c^{(2)} \cdot p_c^{(2)}, \tag{5}$$

where $n_c^{(1)}$ and $n_c^{(2)}$ are the highest class count and the second highest count in the cell c respectively, $p_c^{(1)}$ is the probability that $n_c^{(1)} + Z_{(1)} \geq n_c^{(2)} + Z_{(2)}$ and $p_c^{(2)} = 1 - p_c^{(1)}$.

We experimentally study the correlation between the grid quality function (Eq. 1) and its approximation (Eq. 5) over 4 multiclass real datasets (Adult-Multiclass, Bank-Multiclass, US-Multiclass, BR-Multiclass) and 5 privacy budgets (0.05, 0.1, 0.2, 0.5, 1.0), 20 cases in total. For each of cases, we generate 10K candidate grids and compute their grid quality scores by both the grid quality function (Eq. 1) and its approximation (Eq. 5). Figure 3 is the scatterplot of these grid quality scores. We also plot the linear least square fit line to show the tend of the point distribution. Over the 20 cases, the average Pearson correlation coefficient between the grid quality measure (Eq. 1) and its approximation (Eq. 5) is 0.936 with standard deviation 0.026. Therefore, we can see that the simplified multiclass quality function (Eq. (5) is highly correlated with the original one and can well represent the grid quality in multiclass setting.

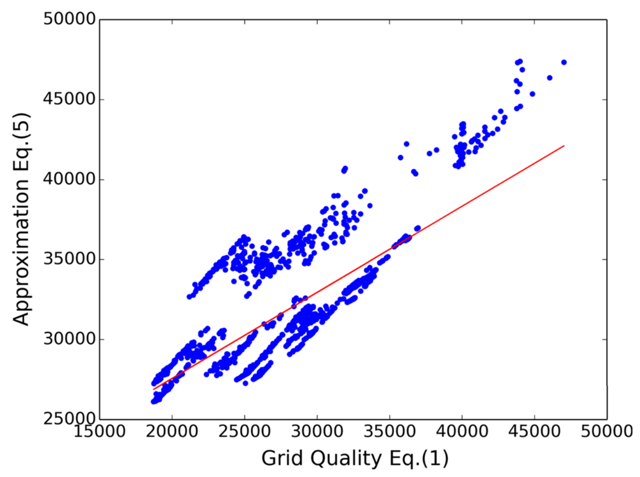

Fig. 3 Scatterplot to illustrate the correlation between grid quality scores computed by grid quality (Eq. 1) and its approximation (Eq. 5) The red line is the linear least square fit for all the points to represent the trend. Average Pearson correlation coefficient is 0.936 with standard deviation 0.026

Lemma 2 *For any $\epsilon > 0$, the global sensitivity of the approximation of the grid quality function for multiclass classification* (Eq. 5) *is bounded by 1.1, that is, $\Delta_{gq} \leq 1.1$.*

The proof is deferred to the "Appendix" section.

4.4 Candidate grids enumeration

PrivPfC takes as input Θ, the maximum number of candidate grids, and generates a pool of at most Θ candidate grids. We also limit the number of cells in each candidate grid, to prevent the average counts in grid cells from being dominated by the injected noises. More specifically, we limit the maximum allowed number of cells in any candidate grid g to be

$$\tau = 0.2\hat{N}\epsilon_{\text{pert}}, \tag{6}$$

where ϵ_{pert} is the privacy budget reserved for adding Laplace noises to the histogram, and $\hat{N}$ is a noisy estimate of the total number of records. The threshold τ is computed by assuming that tuples in the given dataset are of uniform distribution. Then, on average the number of tuples in a cell is equal to $\frac{\hat{N}}{\tau}$. Let $\mathsf{Lap}\left(\frac{1}{\epsilon_{\text{pert}}}\right)$ be the variable of Laplace noises added to cells, where ϵ_{pert} is the privacy budget used to generate the perturbation noises. We require that the noise-to-signal ratio to be less than 20%, that is

$$\mathbb{E}\left[\left|\mathsf{Lap}\left(\frac{1}{\epsilon_{\text{pert}}}\right)\right|\right] \leq 20\% \cdot \frac{\hat{N}}{\tau}. \tag{7}$$

Since $\mathbb{E}\left[\left|\mathsf{Lap}\left(\frac{1}{\epsilon_{\text{pert}}}\right)\right|\right] = \frac{1}{\epsilon_{\text{pert}}}$, we can set the grid size threshold, τ, to be $0.2\hat{N}\epsilon$.

This ensures that the average noise magnitude is no more than the 20% of the average cell count. This non-dominating rule has been used in several differentially private data publishing works [27,37].

PrivPfC generates candidate grids by a level-wise search. It starts from the most general grid, $\langle 1, 1, \ldots, 1\rangle$, where the whole domain is a single cell, and first generates L_1, the list of all grids that have a single attribute going beyond the top level, then generates L_2, the list of all grids that have exactly two attributes going beyond the top level, and so on. It will include a grid as a candidate only when the grid includes no more than τ cells. It stops when either it has included all grids with no more than τ cells, or it has included Θ grids.

The choice of Θ depends on the amount of computing resources one is willing to spend. When Θ is too large, one runs out of candidate grids that have at most τ cells, and increasing Θ further would not increase the size of the pool.

4.5 Putting things together for PrivPfC

Now we are able to put all the pieces together for the PrivPfC method. The algorithm is given in Algorithm 2. PrivPfC takes as inputs D (dataset), T (set of taxonomy hierarchies of predictor variables), ϵ (total privacy budget) and Θ (maximum gird pool size). It returns a noisy histogram which can be used to synthetically generate dataset for classification. PrivPfC has three main steps: (1) Enumerate candidate grids (Line 5); (2) Privately select grid (Line 6); (3) Publish noisy counts (Line 7). Before these steps, it computes the noisy version of dataset size and uses it to compute the grid size threshold according to Eq. 6. We divide the privacy budget into three portions: (1) $3\%\epsilon$ is used to privately estimate the dataset size, (2) $37\%\epsilon$ is used for selecting grid (Function selectGrid) and (3) $60\%\epsilon$ is used for publishing noisy counts (Function perturbHistogram). The enumeration step does not access the dataset D and does not consume any privacy budget. The rationale behind this privacy budget allocation is threefold: (1) The noisy dataset size is used for computing grid size threshold which does not need to be very accurate and can tolerate changes within a large range, e.g., one order of magnitude. Therefore, we only need to allocate a small portion of privacy budget to get the noisy dataset size; (2) there are always a couple of high-quality grids among all generated candidate grids. Selecting any of them can offer good enough classification accuracy. Therefore, this grid selection step does not need large amount of privacy budget; (3) the injected noises for perturbing histogram directly affect the quality of the data and allocating more privacy budget can directly boost the classification accuracy. The privacy budget allocation is the hyperparameter of the PrivPfC algorithm. The current configuration was not finely tuned and may not be optimal. The optimal privacy budget allocation depends on characteristics of the dataset D, the generated grid candi-

Algorithm 2 PrivPfC: Differentially Private Data Publication for Classification

Input: D: dataset, T: set of taxonomy hierarchies of predictor variables, ϵ: total privacy budget, Θ: maximum grid pool size.

Algorithmic Parameters: $\alpha_{\text{size}} + \alpha_{\text{sel}} + \alpha_{\text{pert}} = 1$ decides what proportions of the privacy budget are allocated to the different steps. We use $\alpha_{\text{size}} = 0.03, \alpha_2 = 0.37, \alpha_3 = 0.6$.

```
1: function PrivPfC($D$, T, $\epsilon$, $\Theta$)
2:     $\epsilon_{\text{size}} \leftarrow \alpha_{\text{size}}\epsilon$, $\epsilon_{\text{sel}} \leftarrow \alpha_{\text{sel}}\epsilon$, $\epsilon_{\text{pert}} \leftarrow \alpha_{\text{pert}}\epsilon$
3:     $\hat{N} \leftarrow |D| + \text{Lap}(1/\epsilon_{\text{size}})$    ▹ Noisy estimation of dataset size
4:     $\tau \leftarrow 20\% \cdot \hat{N} \cdot \epsilon_{\text{pert}}$    ▹ Grid size threshold
5:     G $\leftarrow$ Enumerate(T, $\Theta$, $\tau$)
6:     $g \leftarrow$ selectGrid($D$, G, $\epsilon_{\text{sel}}$)
7:     $H \leftarrow$ perturbHistogram($D$, T, $g$, $\epsilon_{\text{pert}}$)
8:     return $H$
9: end function

10: function selectGrid($D$, G, $\epsilon_{\text{sel}}$)
11:     for $i = 1 \to |\text{G}|$ do
12:         $q \leftarrow \text{gq}(\text{G}_i)$ ▹ Compute the grid quality for the $i$'th grid $\text{G}_i$, gq is defined in Eq. (1) and Eq. (5)
13:         $p_i \leftarrow e^{\frac{q\epsilon_{\text{sel}}}{2\times 1.1}}$    ▹ $\Delta_{\text{gq}}$ is bounded by 1.1 by Lemma 1 and 2
14:     $g \leftarrow$ sample $\text{G}_i$ with prob $\frac{p_i}{\sum_i p_i}$
15:     return $g$
16: end function

17: function perturbHistogram($D$, T, $g$, $\epsilon_{\text{pert}}$)
18:     $H \leftarrow$ Initialize($g$, T)
19:     Given grid $g$ and taxonomy hierarchies T, generate list of cells, $\mathcal{C}_g$, defined by grid the $g$
20:     for each cell $c \in \mathcal{C}_g$ do
21:         for each class label $j$ do
22:             $H(c, j) \leftarrow \lfloor \text{count}(D, c, j) + \text{Lap}(1/\epsilon_{\text{pert}}) \rfloor$
23:     return $H$
24: end function

25: function Enumerate(T, $\Theta$, $\tau$)
26:     $L_0 \leftarrow \{\langle 1, 1, \ldots, 1 \rangle\}$
27:     $count \leftarrow 0$
28:     for $k = 0 \to |\text{T}| - 1$ do
29:         $L_{k+1} \leftarrow \{\}$    ▹ List of grid with $k+1$ attributes
30:         for each grid $g \in L_k$ do
31:             for $j = 1 \to |\text{T}|$ do
32:                 if $g_j = 1$ then  ▹ If the $j$'th attribute of grid $g$ has not been generalized
33:                     for $i = 2 \to \text{height}(\text{T}_j)$ do
34:                         $new\_g = \text{clone}(g)$
35:                         $new\_g_j = i$
36:                         if size($new\_g$) $\leq \tau$ then
37:                             $L_{k+1} = L_{k+1} \cup \{new\_g\}$
38:                             $count \leftarrow count + 1$
39:                             if $count \geq \Theta$ then go to 41
40:         if $L_{k+1} == \{\}$ then go to 41
41:     return $\bigcup_{j=1}^{|\text{T}|} L_j$
42: end function
```

Other functions used in the algorithm:

```
43: Initialize($g$, T): Initialize histogram defined by the grid $g$ and taxonomy hierarchies T
44: count($D$, $c$, $j$): Count the number of points in cell $c$ with class label $j$
45: size($g$): the number of cells in the grid defined by $g$ (Definition 2)
46: height($\text{T}_j$): the number of levels of the $j$'th taxonomy hierarchy.
```

dates and the total privacy budget. Given a dataset, it is still an open problem to automatically find the optimal budget allocation satisfying differential privacy while preserving high utility.

Theorem 1 *PrivPfC in Algorithm 2 satisfies ϵ-differential privacy.*

The proof of Theorem 1 straightforwardly follows the primitives and composability properties of differential privacy as shown in Sect. 2. In particular, the dataset size estimation step (Line 3) consumes $3\%\epsilon$, the grid selection step (Line 6) consumes $37\%\epsilon$ and the histogram perturbation step (Line 7) consumes $60\%\epsilon$. The candidate grids enumeration step (Line 5) does not access to the dataset. Therefore, as all data-dependent steps satisfy differential privacy, by the sequential composition property of differential privacy, the PrivPfC algorithm satisfies ϵ-differential privacy.

Time complexity. The most time-consuming step of the algorithm is that of computing the quality of all candidate grids (Line 6), which considers at most Θ candidate grids. Suppose the dataset size is N, computing the quality of one candidate grid takes time $O(N)$ and therefore selecting the grid takes a total time $O(N \cdot \Theta)$. Once a grid is selected, only a single pass over the dataset is needed to do the perturbation (Line 17). Hence, the total running time for PrivPfC is $O(N \cdot \Theta)$.

5 Experiment

5.1 Experimental settings

Evaluation methodology We evaluate the performance of PrivPfC and other competing methods as shown in Table 2 on three different classification tasks: building the CART decision tree classifier, building the support vector machine (SVM) classifier with radial basis kernel and building the logistic regression classifier. For all the experiments, we vary ϵ from 0.05 to 1.0. Similar to the experiment settings of [14,26,34], under each privacy budget, we execute tenfold stratified cross-validation to evaluate the misclassification rate of the methods in Table 2. In this tenfold stratified cross-validation, the dataset is randomly partitioned into 10 subsets, where each subset size is (nearly) equal and preserves the percentage of samples for each class. We then run the experiments for 10 times—for each time, 9 subsets are taken as training data, and the remaining 1 subset is for testing/verification. After running the experiments for 10 times, each subset is used for testing one and only once. For each train-test pair, we use the training data to privately compute a classifier or privately publish the data and build classifier on it. We evaluate classifier's accuracy on the testing data, which is disjoint from the training data. We repeat the process 10 times for each train-test pair. We report the average measurements over the 10 runs and the tenfold cross-validations.

The implementation and experiments of PrivPfC were done in Python 2.7, and all experiments were conducted on an Intel Core i7-3770 3.40GHz PC with 16GB memory.

For methods that output a classifier, i.e., DiffPC-4.5, PrivateERM, PrivGene, PrivLocal and FunctionalMechanism, we use parameters suggested by the corresponding papers. For other data publishing methods, i.e., PrivPfC, PPH, DiffGen, and PrivBayes, we generate private synthetic datasets and then use standard implementations of classification methods on these datasets. To evaluate their performance in terms of the decision tree model, we use the rpart [33] library to build decision trees on synthetic datasets. For evaluation in terms of SVM model, we use the LibSVM package [5]. For evaluation in terms of logistic regression, we use R's glm (generalized linear model) function. When comparing different approaches, we use the same sets of parameters for these classifiers.

We consider two baselines—*Majority* and *NoiseFree*. *Majority* is the misclassification rate by majority voting on the class attribute, which predicts each test case with the majority class label in the train dataset. *NoiseFree* is the misclassification rate of a decision tree, an SVM classifier or a logistic regression classifier built on the true data. We expect that a good algorithm to perform significantly better than *Majority*, and gets close to *NoiseFree* as ϵ increases.

Datasets We experimented with 8 real datasets and 2 sets of synthetically generated datasets. They are summarized in Table 1.

The first real dataset is the Adult dataset from the UCI machine learning repository [1]. It contains 6 numerical attributes and 8 categorical attributes, and is widely used for evaluating the performance of classification algorithms. After removing missing values, the dataset contains 45,222 records. We create a multiclass version of the Adult dataset, called Adult-Multiclass, by using the 3-valued marital status attribute as the class attribute.

Table 1 Dataset characteristics

Dataset	# Dim	# Num	# Cate	# Rec	# Class	Classification task
Adult	15	6	8	45,222	2	Determine whether a person makes over 50K a year
Adult-Multiclass	15	6	8	45,222	3	Determine the three classes marital status of a person
Bank	21	10	10	41,188	2	Determine whether the client subscribed a term deposit
Bank-Multiclass	21	10	10	41,188	3	Determine three types of outcome of the previous campaign
US	47	15	31	39,187	2	Determine whether a person makes over 50K a year
US-Multiclass	46	15	30	39,187	4	Determine the four types of school attended by a person
BR	43	14	28	57,333	2	Determine whether a person makes over 300 per month
BR-Multiclass	43	14	28	57,333	4	Determine the four types of employment status of a person
Synthe-binary	21	20	0	{5K, 10K, 20K, 50K,100K, 200K, 500K}	2	Determine class label {0, 1} of a point
Synthe-multi	21	20	0	{5K, 10K, 20K, 50K,100K, 200K, 500K}	2	Determine class label {0, 1, 2, 3, 4} of a point

Table 2 Summary of differentially private classification methods

Methods	Description
Data publishing	
PrivPfC	Our proposed method
PrivPfC-SelNF	Our proposed method with noise-free grid selection
DiffGen [26]	Private data release for classification via recursive partitioning
DiffGen-Struct-NF	DiffGen with noise-free partitioning procedure
PrivBayes [37]	Private Data Release via Bayes network
PrivBayes-Struct-NF	PrivBayes with noise-free network learning procedure
PPH [34]	Private data release for classification by projection and perturbation
Classifier-outputting	
DiffPC-4.5 [14]	Privately construct C4.5 decision tree classifier
PrivGene [38]	Private model fitting based on genetic algorithms
PrivLocal [38]	Private local search algorithm
FunctionalMechanism [39]	Private model fitting by perturbing the fitting function
PrivateERM [7]	Private classifier construction based on empirical risk minimization

The second dataset is the Bank marketing dataset from the same repository. It contains 10 numerical attributes and 10 categorical attributes ON 41,188 individuals. The multiclass version of the Bank dataset is created by using 3-valued the poutcome attribute as the class attribute. The third is the US dataset from the *Integrated Public Use Microdata Series* USA (IPUMS-USA) [29]. It has 39,187 US census records in 2010, with 15 numerical attributes and 31 categorical ones. The multiclass version of the US dataset is created by using the 4-valued SCHLTYPE attribute as the class attribute. We remove one categorical attribute which is highly correlated with the SCHLTYPE attribute in the multiclass version of US dataset. The fourth is the BR dataset from the Integrated Public Use Microdata Series International (IPUMS-International) [25], which contains 57,333 Brazil census records in 2010 and has 14 numerical attributes and 28 categorical ones. The multiclass version of the BR dataset is created by using the 4-valued EMPSTAT attribute as the class attribute.

We generate 2 sets of synthetic datasets for evaluating algorithms' performances on varying dataset sizes. The first set of synthetic datasets, which we call Synthe-binary, contains 7 datasets with binary class labels and with size 5K, 10K, 20K, 50K, 100K, 200K and 500K, respectively. The second set of synthetic datasets, which we call Synthe-multi, contains 7 dataset with 5 class labels and with size 5K, 10K, 20K, 50K, 100K, 200K and 500K, respectively.

Taxonomy hierarchies For the Adult and Adult-Multiclass datasets, we use the same taxonomy hierarchies as those in DiffGen [26]. For the remaining 6 datasets, we create taxonomy hierarchies as follows. For numerical attributes, we partition each domain into equal size bins and build hierarchies over them. For categorical attributes, we build taxonomy hierarchies by considering the semantic meanings of the attribute values.

5.2 Competing methods

We compare PrivPfC with 8 state-of-the-art methods in terms of misclassification rate. These include 3 data publishing methods that publish either a noisy histogram or a noisy Bayesian network, which can be used to generate a synthetic dataset: DiffGen [26], PrivBayes [37], and PPH [34]; and 5 methods that directly output a classifier, PrivLocal [38], PrivGene [38], FunctionalMechanism [39], DiffPC-4.5 [14], and PrivateERM [7]. Table 2 summarizes the competing methods mentioned in this paper.

DiffGen [26] also uses taxonomy and publishes a noisy histogram. However, it chooses the grid in a way different from PrivPfC. In DiffGen, one iteratively selects one attribute at a time for specialization, using the exponential mechanism. The quality function suggested in [26] aims to maximize the number of records that have the majority class label in all cells.

The number of specialization steps is an important parameter and is an input to the algorithm. As suggested in [26], we set the number of specialization steps to be 10 for the Adult dataset, Adult-Multiclass and the bank dataset. For the US dataset and its multiclass version, we set the number to be 6. For the BR dataset and its multiclass version, we set it to be 8. Because beyond these numbers, the DiffGen implementation runs into memory problems, because the taxonomy trees for these datasets have larger fan-outs.

PPH [34] also publishes a noisy histogram. It uses the exponential mechanism to select k attributes, using a quality function that maximizes the discernibility score regarding the label attribute. The grid is determined by the k attributes. For each categorical attribute, the full domain is used. For a numerical attribute, it uses the formula proposed in Lei [23] to decide how many bins to discretize the attribute domain.

PrivBayes [37] publishes a noisy Bayesian network. It determines the structure of a Bayesian network by first randomly selecting an attribute as the first node, and then iteratively selecting another attribute to create a new node and up to k already created nodes as the new node's parent nodes. PrivBayes is only applicable on binary classification. After the structure is determined, PrivBayes perturbs the marginals needed for computing the conditional distributions. The performance of the PrivBayes algorithm depends on k. We set $k = 3$ for the Adult dataset and the Bank dataset, which is the same as the one used in [37]. For the US and BR datasets, which were not used in [37], setting $k = 3$ runs out of memory in our experiments because these datasets have more attributes; we set $k = 2$ for them.

Classifier-outputting methods *PrivGene* [38] is a general-purpose private model fitting framework based on genetic algorithms, which can be applied to SVM classification and logistic regression. *PrivLocal* is a differentially private local search algorithm. *DiffPC-4.5* [14] outputs a C-4.5 decision tree classifier differentially privately. *PrivateERM* [7] outputs an SVM classifier by injecting noise into the risk function first and then optimizing the perturbed risk function.

The source codes of DiffGen, PrivBayes, PPH, DiffPC-4.5, PrivLocal were shared by authors of corresponding papers. The source code of PrivateERM was shared by the authors of PrivGene. We implement the PrivGene algorithm by strictly following the paper [38].

5.3 Comparison with existing solutions

For each classification method, we compare PrivPfC with $\Theta = 10{,}000$, with three existing data publishing methods DiffGen, PrivBayes, PPH and any classifier-outputting method that can be applied to this classification method. We note that PrivBayes is not designed to be optimized for a single classification task, thus in some sense is not expected to perform well.

Figure 4 reports the average misclassification rates and the corresponding standard deviations for the decision tree classification.

Clearly, PrivPfC has the best performance in most cases, followed by DiffGen, PPH, DiffPC-4.5 and PrivBayes. The performance of PrivPfC is also the most robust, as can be seen from the fact that the standard deviations of its misclassification rates are always the lowest.

Figure 5 shows similar experimental results for SVM classification. PrivPfC has the best performance, followed by DiffGen, PrivLocal, PPH, PrivateERM, PrivBayes and PrivGene. PrivGene performs the worst, because the crossover operation in each iteration significantly destroys the structure selected SVM parameter by misaligning the parameter values to their corresponding dimensions. On the other hand, PrivLocal only uses perturbation to generate offsprings and the structure of SVM parameters can be largely kept. This result also confirms our remarks on the effectiveness of PrivGene.

We can also see that as the increase of privacy budget from 0.05 to 1.0, PrivPfC, DiffGen and PPH's performances are not affected too much, PrivBayes's performances sometimes change a lot, and PrivLocal and PrivateERM's performance constantly improve. The reason is threefold. (1) PrivPfC, DiffGen and PPH allocate a portion of privacy budget to determine the histogram structure to partition the data. And there are always a couple of structures which result in close classification performance. With a small change of privacy budget, these algorithm might still select a structure which has close performance with the previous one. (2) For PrivBayes, it uses privacy budget to heuristically determine the degree of the Bayesian network structure, where more privacy budget results in larger degree. However, PrivBayes has to round the computed degree to the closest integer. Therefore, as we see in Fig. 5, especially figures for US and BR datasets, when the privacy budget reaches 0.2, the misclassification error of PrivBayes is significantly reduced and then only gets minor improvements as the privacy budget increases until reaching the next point to change the rounded integer degree. (3) For PrivateERM, since noises are directly injected into the objective function, less noise can make the noisy objective function being closer to the true version, which improves PrivateERM more directly.

Figure 6 reports the experimental results on logistic regression. Overall, PrivPfC has the best performance, followed by PrivLocal, DiffGen, PPH, PrivBayes, FunctionalMechanism and PrivGene. PrivGene performs the worst again. Note that, in the US and BR dataset, when the privacy budget is large, PrivLocal outperforms PrivPfC with a slight advantage. This is because PrivLocal has a tighter sensitivity bound when applying to logistic regression. Besides, when the privacy budget is large, PrivPfC selects a subset of features to build histogram, whereas the PrivLocal can use the full set of dimensions to build the classifier.

While the idea of making a genetic programming algorithm differentially private is interesting, the effectiveness of the PrivGene algorithm is questionable for several reasons. First, the crossover operation often does not result in competitive candidates. Second, with crossover and mutation, the convergence rate is low, which means that a larger number of iterations are needed. Third, for each iteration, the algorithm requires making a large number of selections, with every single one of them consuming some privacy budget.

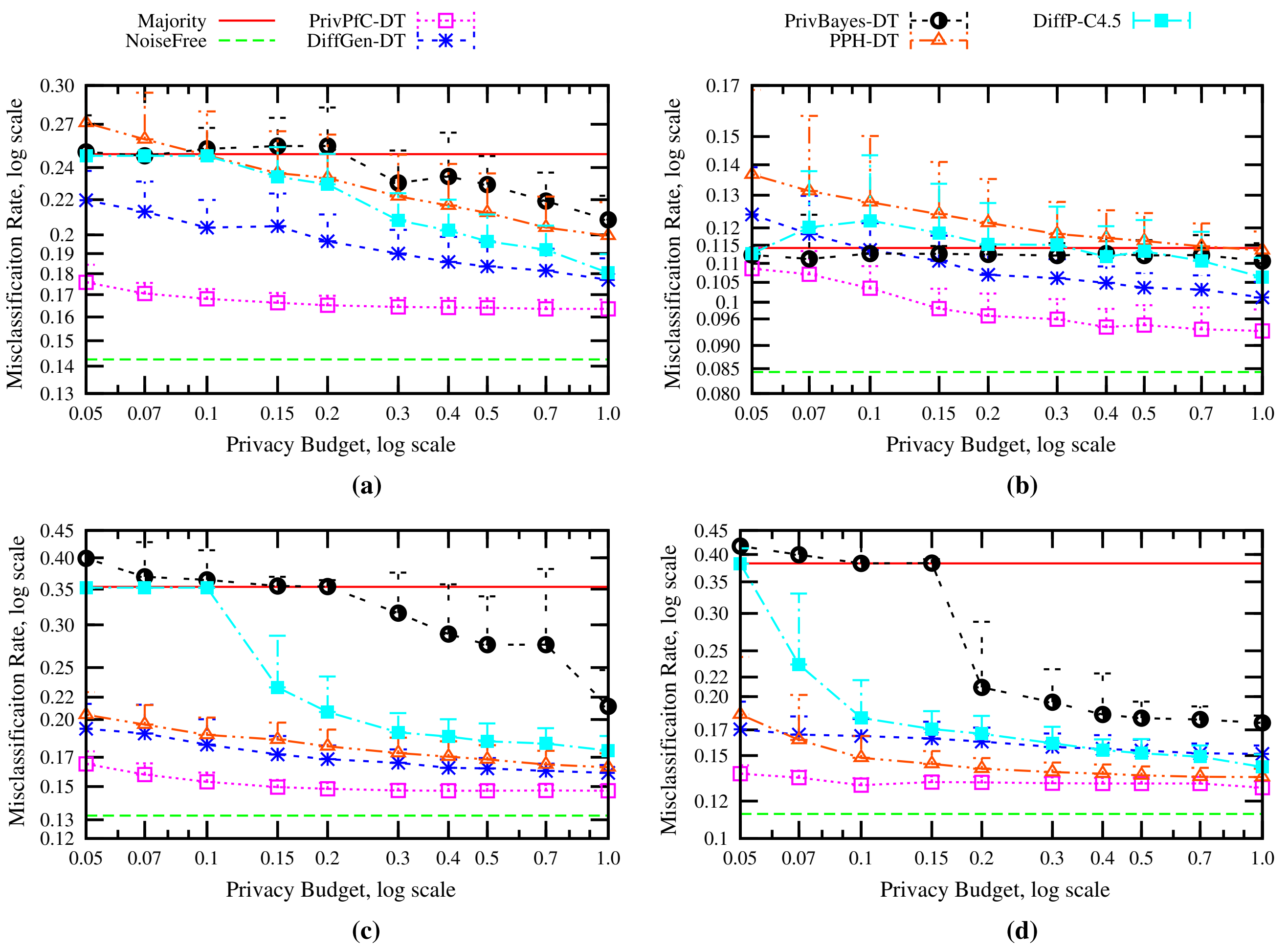

Fig. 4 Comparison of PrivPfC, DiffGen, PrivBayes, PPH and DiffPC-4.5 by decision tree classification. DT: evaluated by decision tree classification. **a** Adult, **b** Bank, **c** US, **d** BR

As the increase of privacy budget from 0.05 to 1.0, we also get similar observations as those in Fig. 5. In this figure, FunctionalMechanism's performance constantly improves. This is because in FunctionalMechanism, the noises are injected into the coefficients of the approximation polynomial of the objective function. Therefore, the injected noises have direct impact to the classification accuracy.

Furthermore, when comparing PrivPfC's performances under three classification tasks in Figs. 4, 5 and 6, one can see that the gap between PrivPfC and the noise-free baseline on decision tree classification is relatively larger than those for SVM and logistic regression classification. This is mainly because PrivPfC only selects informative attributes and discards other non-informative attributes Given data under these informative attributes, linear models like SVM and logistic regression can make accurate classification, while decision tree models always need information from other less informative attributes to achieve similar performance.

Comparison on multiclass classification We compare five approaches: PrivPfC, DiffGen, PPH, PrivLocal and PrivGene on multiclass classification on 4 real datasets, Adult-Multiclass, Bank-Multiclass, US-Multiclass and BR-Multiclass. The evaluations of the three non-interactive data publishing methods, PrivPfC, DiffGen and PPH are done by the decision tree classification, since these methods privately generate synthetic datasets and decision tree can naturally supports multiclass classification. The PrivLocal and PrivGene methods only produce one classifier, SVM or Logistic regression at a time. We therefore use the One-vs.-rest approach [3] to reduce the multiclass classification problem into the binary classification problem and use ϵ/k budget to train each classifier, where k is the number of classes. Figure 7 shows the experimental results. PrivPfC is again the winner in most cases.

5.4 Varying parameters in PrivPfC

We now explore the effect of changing Θ, the maximum grid pool size and the effect of using different privacy budget allocation plans in PrivPfC. Figure 8 reports the results of

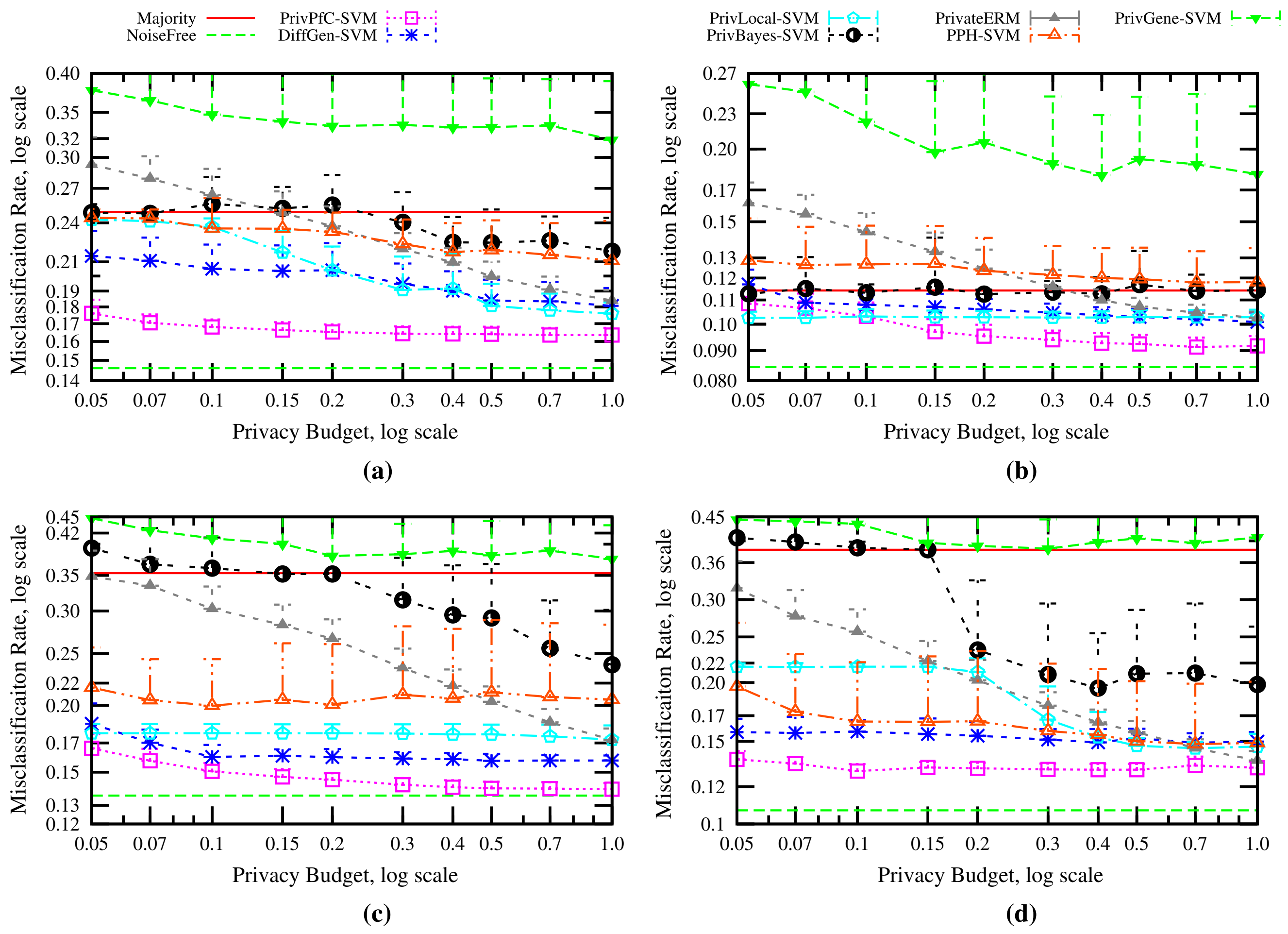

Fig. 5 Comparison of PrivPfC, DiffGen, PrivBayes, PPH, PrivGene and PrivateERM by SVM classification. SVM: evaluated by SVM classification. **a** Adult, **b** Bank, **c** US, **d** BR

PrivPfC's performance under three Θ values, 100, 10,000 and 200,000. The evaluation is done on the BR dataset with two privacy budgets, 0.05 and 0.5. We can see that with the increasing of the maximum pool size, PrivPfC's performance gets significant improvement from $\Theta = 100$ to $\Theta = 10{,}000$. When setting Θ to the largest value, 200,000, PrivPfC also gets a small amount of improvement.

PrivPfC distributes the privacy budget among three steps, size estimation, grid selection and perturbation, with the ratio 3–37–60%.

We have experimentally evaluated other privacy budget allocation ratios, ranging from 20% to 60%, for the grid selection and perturbation steps respectively. We have found that the differences among different budget allocations are minor, so long as the last step receives at least 30% of the privacy budget. Figure 9 compares the performance of DiffGen, PrivPfC with privacy budget allocation 3–37–60% and PrivPfC with privacy budget allocation ratio 3–10–87%. Both PrivPfC versions use $\Theta = 10{,}000$. We can see that PrivPfC-3–10–87% performs reasonably well, and in fact slightly better than the standard PrivPfC when $\epsilon \geq 0.2$.

This also shows that our PrivPfC algorithm is robust under different privacy budget allocation plans.

5.5 Analyses of sources of errors

We have seen that PrivPfC outperforms the other data publishing methods such as DiffGen and PrivBayes. The key difference in PrivPfC is that we choose the grid g in a single step, instead of arriving at the final grid through a series of decisions. For example, DiffGen iteratively chooses the attributes and ways to partition them, and PrivBayes iteratively builds a Bayesian network. There are two reasons why such an iterative approach does not perform well. The first is that the decisions made in each iteration may be suboptimal because of the randomization necessary for satisfying differential privacy. The second is that even if the decision made in each iteration is locally optimal, the combination of them is not globally optimal. To see to what extent the latter factor affects accuracy, we consider variants of them, respectively, DiffGen-Struct-NF and PrivBayes-Struct-NF. In these variants, the decisions in each iteration are performed without

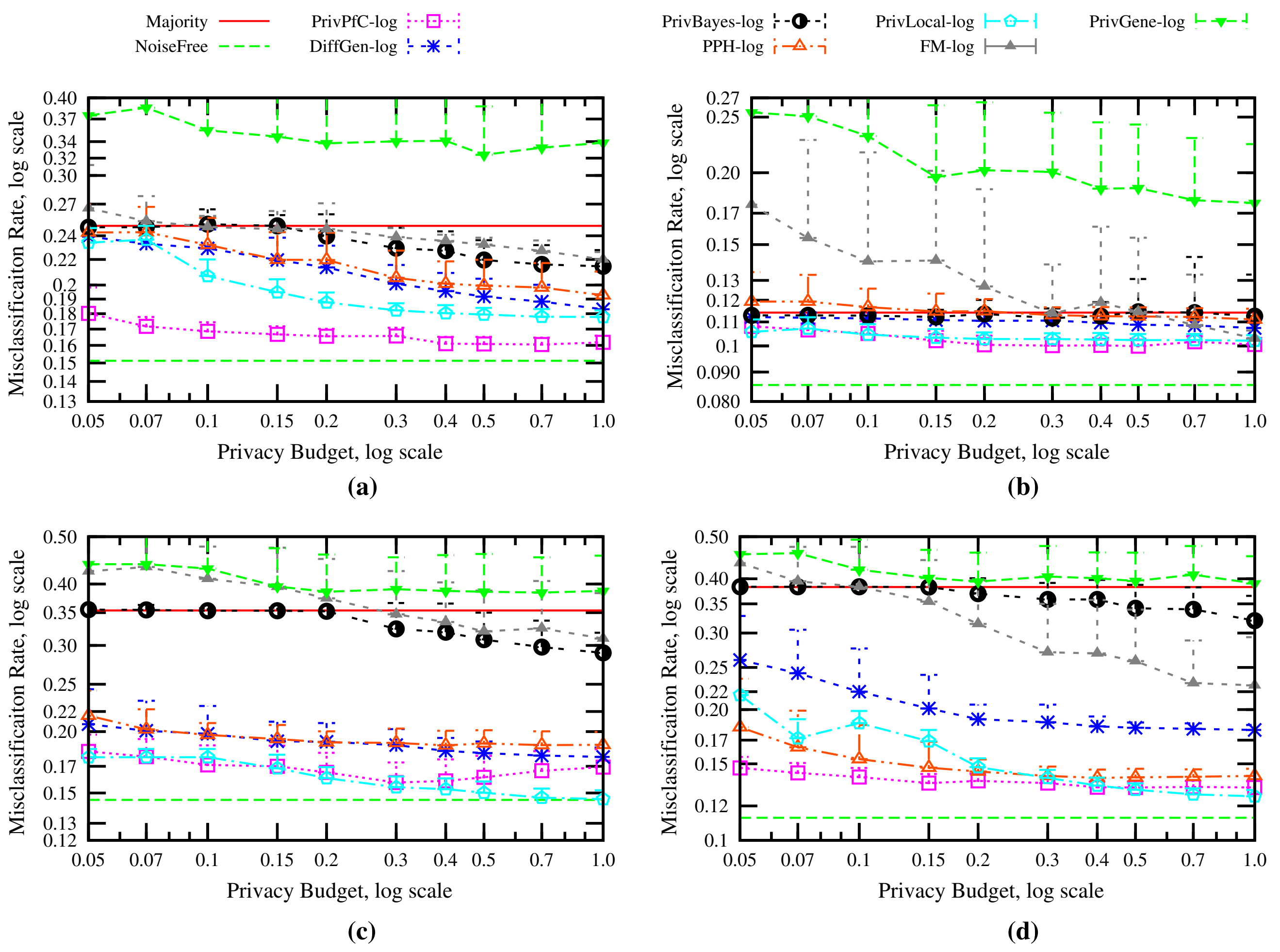

Fig. 6 Comparison of PrivPfC, DiffGen, PrivBayes, PPH and FunctionalMechanism by logistic regression classification. log: evaluated by logistic regression. **a** Adult, **b** Bank, **c** US, **d** BR

any perturbation, but noises are still added when publishing the counts.

We also consider a variant of PrivPfC, called PrivPfC-SelNF, in which the histogram selection step is noise-free. All these variants are not private; they are used to understand the source of errors only.

Figure 10 reports the experimental results of comparing these methods, using the decision tree classifier. We first observe that PrivPfC-SelNF indeed outperforms PrivPfC, although the differences tend to be smaller than the difference between PrivPfC and DiffGen. We also observe that PrivBayes-Struct-NF still performs poorly; in fact, it performs significantly worse than PrivPfC. Again, this is not surprising given that the iterative Bayes network construction approach is not designed to optimize one classification task. Similarly DiffGen-Struct-NF still underperforms PrivPfC. This suggests that the inherent iterative structure of DiffGen is suboptimal. In summary, in releasing data for classification under differential privacy, it is better to determine the structure of the data synopsis in a single step instead of a series of steps.

5.6 Scalability over dimensions, dataset size and runtime

We study the scalability of dimensions of our algorithm as well as our competitors. This experiment is performed on the US dataset. First, we sort all of its predictor variables by their degrees of correlation to the response variable in descending order. The correlation is measured by the χ^2 statistic, which is one of the most effective methods of feature selection for classification [16,36]. We then generate the set of datasets with lower number of dimensions by projecting the US dataset to dimensions defined by the first $d-1$ predictor variables and the response variable, where $d = 10, 15, 20, 25, 30, 35, 40, 47$.

Figure 11 shows the results. We can see that as the increasing of dimensionality, PrivPfC, DiffGen and PPH offer stable classification performance. PrivPfC is still the best among

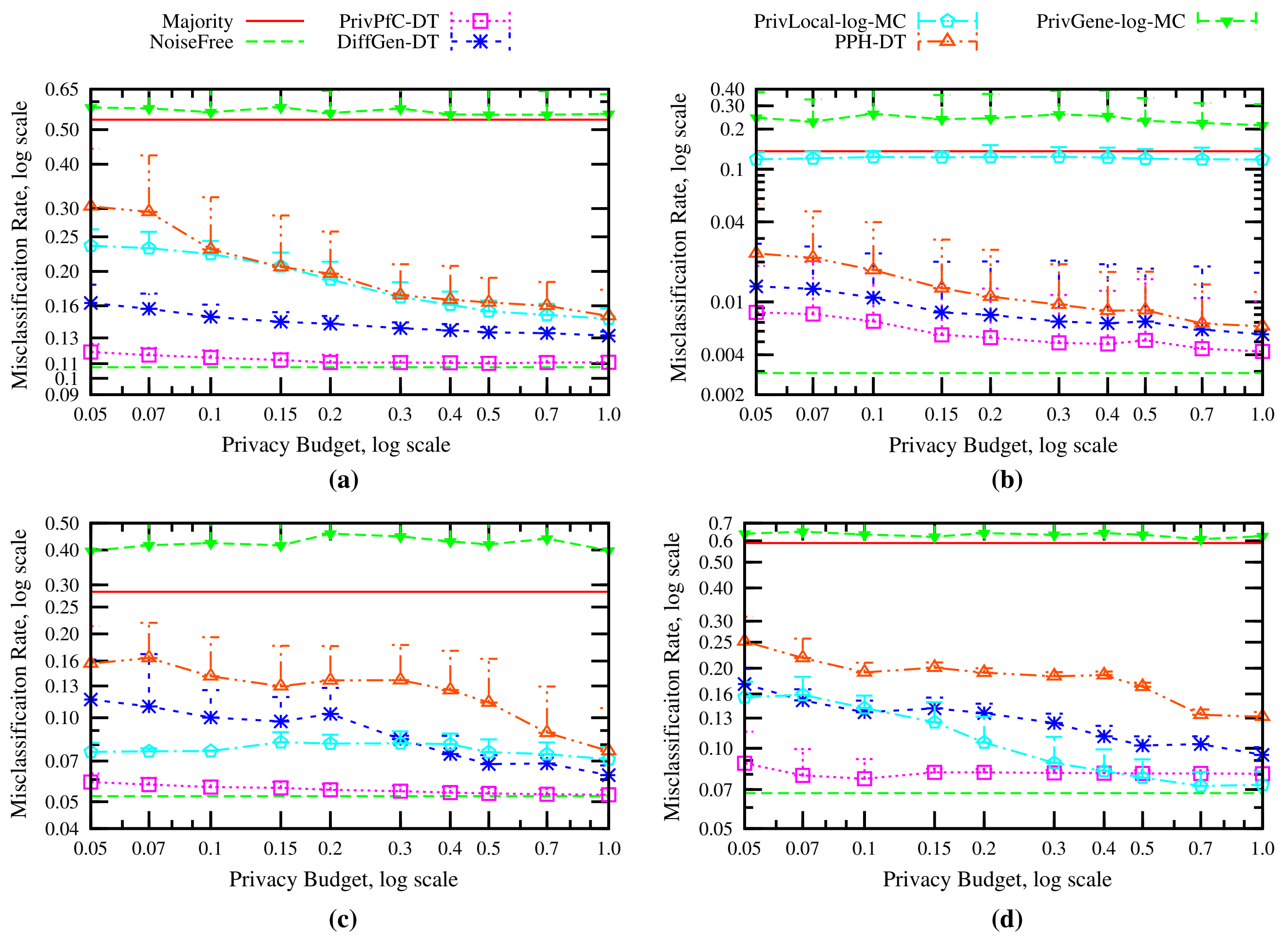

Fig. 7 Comparison of PrivPfC, DiffGen, PPH, PrivLocal and PrivGene by decision tree classification and logistic regression classification on the multiclass datasets. DT: evaluated by decision tree classification, log: evaluated by logistic regression, MC: multiclass classification. **a** Adult-Multiclass, **b** Bank-Multiclass, **c** US-multiclass, **d** BR-multiclass

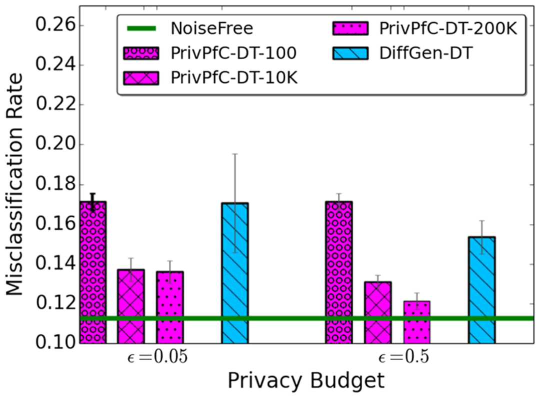

Fig. 8 Varying the maximum pool size Θ on PrivPfC by decision tree classification on the BR dataset. DT: evaluated by decision tree classification. 100, 10K, 200K are values of Θ

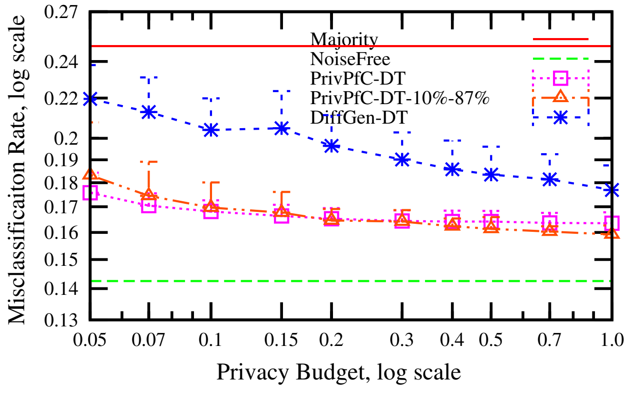

Fig. 9 Comparison of two different privacy budget allocations on PrivPfC by decision tree classification on the Adult dataset. DT: evaluated by decision tree classification. 10–87%: in PrivPfC, allocate 10% of privacy budget to selectGrid step and 87% to perturbHistogram step

them. In essence, all of these three algorithms are trying to construct a histogram structure to partition the dataset by identifying all informative attributes for classification. The number of informative features of the census dataset, e.g., US dataset, is relatively smaller than the total number of dimensions. All of these three algorithm can find most of

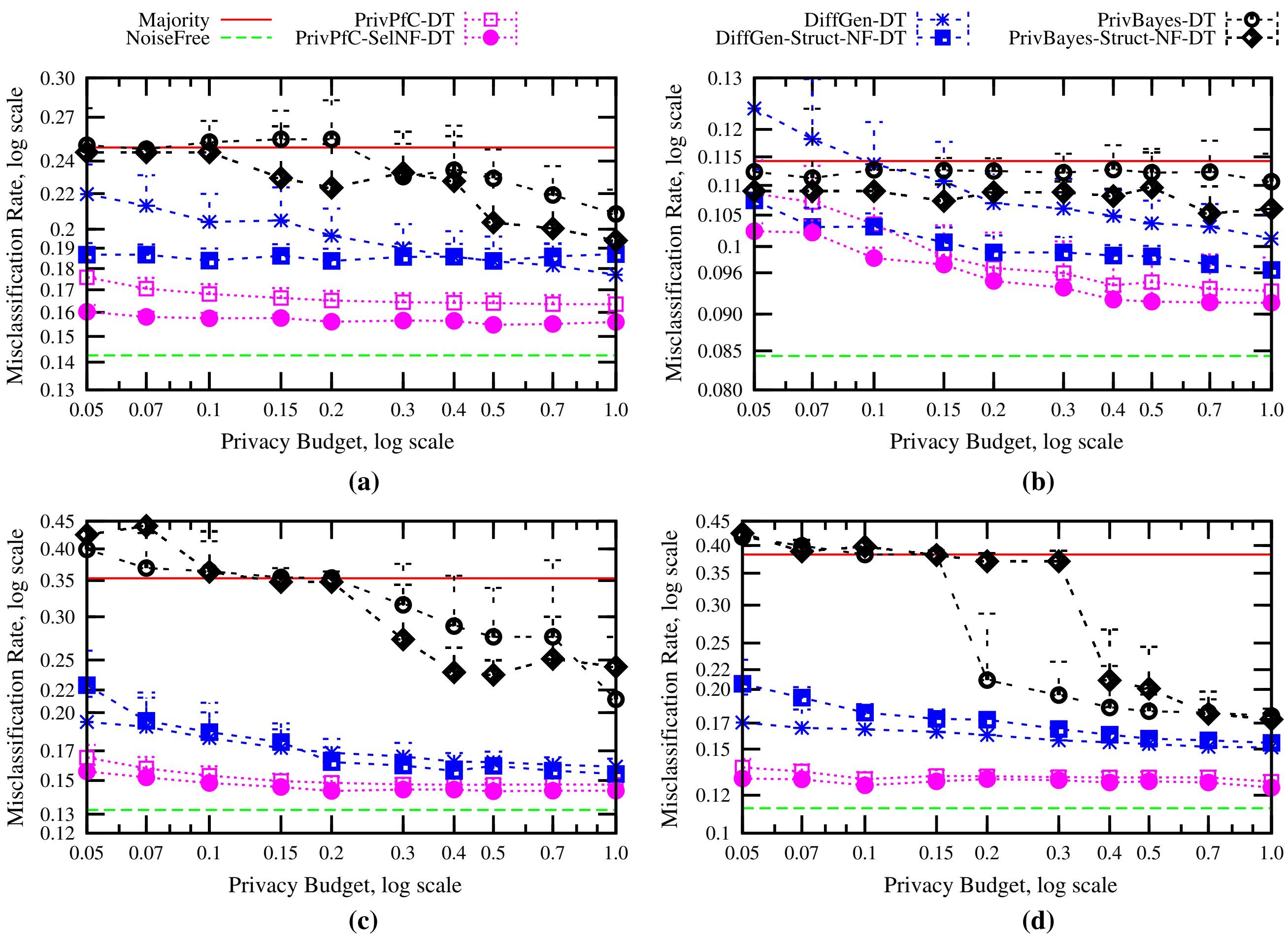

Fig. 10 Analyses of PrivPfC, DiffGen and PrivBayes by decision tree classification. DT: evaluated by decision Tree classification. **a** Adult, **b** Bank, **c** US, **d** BR

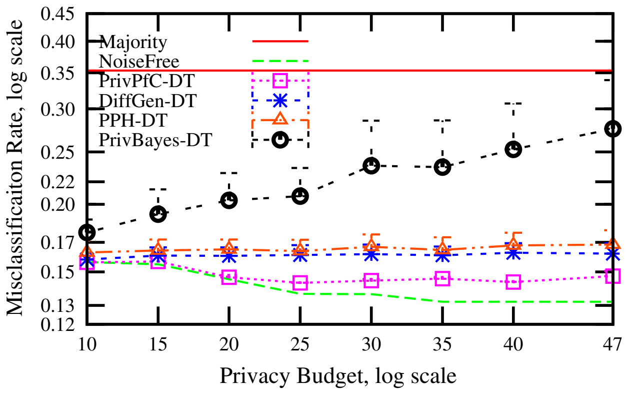

Fig. 11 Comparison of PrivPfC, DiffGen, PrivBayes and PPH by varying dimensions (decision tree classification). $\epsilon = 0.5$. DT: evaluated by decision tree classification

these informative attributes. But among them, PrivPfC does the best job. PrivBayes is the poorest in all cases, and its performance goes worse as the dimensionality increases. This is because in PrivBayes, the magnitude of the noises injected into the marginals of constructed Bayesian network is proportional to the number of dimensions.

We compare the scalability of our algorithm and non-interactive data publishing algorithms, DiffGen, PPH and PrivBayes over datasets with different sizes. We use the set of synthetically generated datasets, Synthe-binary and Synthe-multi, for binary classification and multiclass classification, respectively. Figure 12 shows the comparison results. As the increasing of dataset size, the misclassification rates of all methods decrease. PrivPfC always gives the best performance.

We also compare the running time of 4 data publishing algorithms, PrivPfC, DiffGen, PrivBayes and PPH, on the US dataset with privacy budget 0.05 and 0.5, respectively. Figure 13 shows the comparison results. PrivBayes is the most inefficient one, followed by PrivPfC, PPH and DiffGen. By considering runtime comparison results and accuracy comparison results (Figs. 4, 5, 6, 7) together, we can see that PrivPfC trades more runtime for accuracy improvement. From the runtime comparison result, we can also see that under different privacy budgets, PrivBayes, PrivPfC and PPH have close runtime, while DiffGen needs longer time when the privacy budget gets larger. This is because with more pri-

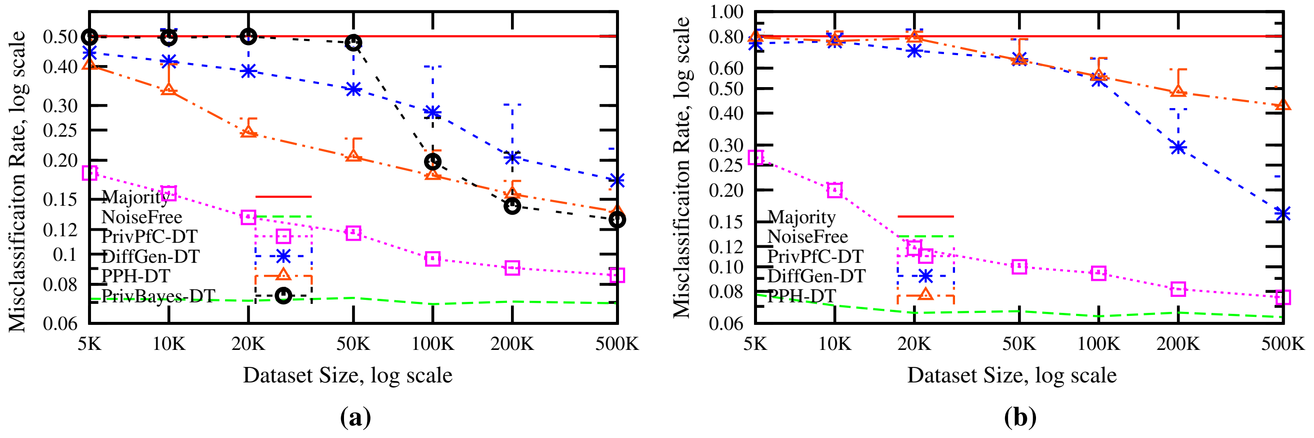

Fig. 12 Comparison of PrivPfC, DiffGen, PrivBayes and PPH by varying dataset size on Synthe-binary datasets and Synthe-multi datasets (decision tree classification). $\epsilon = 0.1$. DT: evaluated by decision tree classification. **a** Synthe-binary, **b** synthe-multi

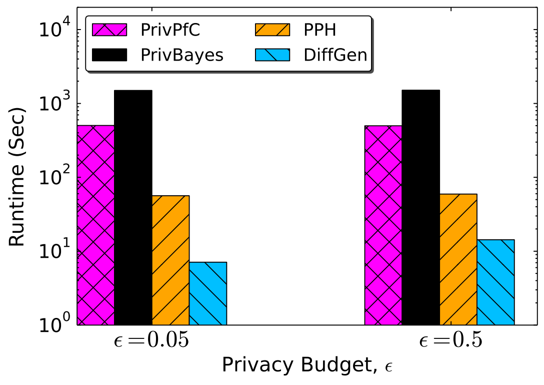

Fig. 13 Runtime comparison of PrivPfC, DiffGen, PPH and PrivBayes by decision tree classification on the US dataset

vacy budget DiffGen is likely to choose finer partitions in attribute taxonomy hierarchy, which results in more time to project data into the partition structure.

For those classifier-outputting methods, DiffPC-4.5 is always the fast approach, since the differentially private decision tree is constructed quickly. Compared with DiffPC-4.5, FunctionalMechanism and PrivateERM have higher running time cost, since they both have to invoke numerical optimizer to optimize the perturbed objective functions. PrivGene and PrivLocal always need larger running time since they need a larger number of iterations.

5.7 Discussion

Running time As shown in Sect. 4.5, PrivPfC has time complexity $O(N \cdot \Theta)$, where N is the dataset size and Θ is the maximum grid pool size. From Figs. 8 and 13, we can see that when using a small Θ value, PrivPfC has close classification accuracy with DiffGen. When increasing Θ values, PrivPfC is able to offer better accuracy, at the cost of increasing running time. In other words, compared with DiffGen, PrivPfC offers an extra "tuning nob," where one can get better utility by using more computational resources. DiffGen and other algorithms do not have such a "tuning nob."

We argue that this "tuning" capability is highly valuable. Many datasets that people want to use are small. Such small datasets create the most serious contentions between privacy and utility, because the impact of noises is larger for smaller datasets. PrivPfC can relax such contention by providing a way to improve utility by spending more computational resources. Also, for small datasets, even the higher computational costs of PrivPfC are easily affordable. We also note that since PrivPfC publishes a noisy histogram. For each dataset, one needs to run PrivPfC only once.

Dimensionality We use datasets of less than 50 dimensions/attributes to compare the scalability over dimensions of our PrivPfC and other competing methods. From Fig. 11, we can see that PrivPfC keeps its superiority on classification accuracy and scales at least as well as other competing methods except the PrivBayes. PrivBayes's classification error grows with the increase of number of dimensions, since it has to perturb marginals with Laplace noise which scales linearly to the number of dimensions.

However, it is an open question how to accurately perform classification while satisfying differential privacy for datasets with hundreds or more dimensions. This challenge is in twofold.

First, for datasets of higher dimensions and more than several informative attributes, PrivPfC need a high Θ value to ensure that the good grids (which partition several informative attributes) are included in the pool of candidates. This makes PrivPfC prohibitively expensive.

Second, given a relational dataset and one class attribute, there are always a few informative attributes which are highly correlated with this class attribute. One data record's class label can be largely determined by values under these attributes. Therefore, from the perspective of dimension reduction, PrivPfC can be seen as projecting the data space into a lower dimension by selecting a set of informative attributes which are highly correlated with the class attribute. As long as the number of informative attributes is not very large, PrivPfC can always work well, even if the total number of attributes is large. On the other hand, if the number of informative attribute is large, selecting a grid to capture all of them and perturbing the selected histogram to get good classification accuracy might be challenging. This is because good grid candidates cannot easily stand out from a very large pool of candidate grids and the exponential mechanism cannot accurately select a good grid from it. Even if the good grid can still be selected, the class counts in each cell of resulted histogram might be very small and are easily dominated by the injected noises.

Privacy budget allocation PrivPfC divides the total privacy budget ϵ into three portions: $3\%\epsilon$ for estimating the total number of records, $37\%\epsilon$ for selecting the grid, and $60\%\epsilon$ for adding noises to the histogram. The choice of how ϵ is partitioned is heuristic in nature. While it is desirable to have a more principled approach to decide how the total privacy budget is partitioned among different steps, unfortunately, the current state of the art in designing differentially private algorithms does not provide the tools to do this. The main challenge lies on the unclear relationship between the classification accuracy and the injected randomness into each data accessing steps.

On generalizability of PrivPfC to other machine learning tasks The data published by our PrivPfC method can be used for performing all existing classification tasks. PrivPfC does not rely on any existing classification algorithm to work. Given a dataset and the class attribute of it, PrivPfC privately releases a synthetic version of the data by selecting a grid structure to partition the data space and perturbing the resulted histogram. This selection is done by privately picking the grid which maximizes the expected number of correctly classified records over all cells in the histogram defined by such grid. When we design the grid quality function in Sect. 4.1, we use the term, histogram classifier, to describe the quality function. Such histogram classifier is actually a theoretical classifier which is always used for modeling the data distribution for classification.

For unsupervised learning tasks, e.g., k-means clustering, we also proposed a method, called Extended Uniform Gridding (EUG) [31,32] to construct noisy histograms to privately publish data for k-means clustering. We showed that EUG has good enough performance for data with number of dimensions from 2 to 10 comparing with other competing methods on differentially private k-means clustering. PrivPfC cannot be applied to this setting, since it is tailored for classification on relational data and assumes the dataset has a class attribute.

6 Conclusion

In this paper, we have introduced PrivPfC, a novel framework for publishing data for classification under differential privacy. As a core part of PrivPfC, we have introduced a novel quality function that enables the selection of a good "grid" for publishing noisy histograms in a single step. We have conducted extensive experiments on 8 real datasets with 3 commonly used classification models, and compared with 8 competing methods. Experimental results show that our approach greatly outperforms several other state-of-the-art methods for private data publishing as well as private classification. This counter-intuitive result points to the future research direction of designing better private classification algorithms by using as few steps as possible, avoiding spreading the privacy budget too thin, perhaps by exploiting the exponential mechanism directly.

Acknowledgements We thank the reviewers for their valuable comments. This paper is based upon work supported by the United States National Science Foundation under grants CNS-1116991 and CNS-1640374 and Key Laboratory on High Performance Computing, Anhui Province, NSFC (61672486, 61672480,11671376), Key Program of NSFC (71631006).

7 Appendix

Lemma 3 gives the distribution of the difference of two i.i.d. Laplace random variables, which will be used in the proof of Lemma 1.

Lemma 3 [20] *Let Z_1 and Z_2 be two i.i.d. random variables that follow the Laplace distribution with mean 0 and scale $\frac{1}{\epsilon}$. Then the density of their difference $Y = Z_1 - Z_2$ is*

$$f_Y(y) = \frac{\epsilon}{4} e^{-\epsilon|y|}(1 + \epsilon|y|) \qquad -\infty < y < \infty,$$

and the corresponding cumulative distribution function is

$$F_Y(y) = \begin{cases} 1 - \dfrac{e^{-\epsilon y}}{2}\left(1 + \dfrac{\epsilon y}{2}\right), & \text{if } y \geq 0, \\ \dfrac{e^{\epsilon y}}{2}\left(1 - \dfrac{\epsilon y}{2}\right), & \text{otherwise.} \end{cases} \tag{8}$$

Proof of Lemma 1

Proof We show the global sensitivity of the grid quality (Eq. 1) in the binary classification setting can be safely bounded by 1.1.

Given a dataset D, and without loss of generality we assume that D has 2 class labels $\{1, 2\}$ and the neighboring dataset of D is $D' = D - t$, where the tuple t falls into cell e and has class label 1. Given a grid g, the quality values of all cells of it are the same except the cell e. In the cell e, we have the number of data points with label 1, $n_e^1, {n'}_e^1$, and that with label 2, $n_e^2, {n'}_e^2$ for D and D', respectively, where $n_e^1 = {n'}_e^1 + 1$ and $n_e^2 = {n'}_e^2$. Each class also has a probability, p_e^i, being the dominant class in cell e, where $i = \{1, 2\}$.

To compute the global sensitivity of the grid quality (Eq. 1), we first present the difference of the grid quality function for the neighboring datasets D and D' in terms of n_e^1, n_e^2, p_e^1 and p_e^2. Then we show that the difference can be bounded by 1.1.

The global sensitivity of the grid quality function for binary classification can be computed by,

$$\begin{aligned}
\Delta_{\mathsf{gq}} &= |\mathsf{gq}(D, g) - \mathsf{gq}(D', g)| \\
&= \left|\left(n_e^1 p_e^1 + n_e^2 p_e^2\right) - \left({n'}_e^1 {p'}_e^1 + {n'}_e^2 {p'}_e^2\right)\right| \\
&= \left|\left(n_e^1 p_e^1 + n_e^2 p_e^2\right) - \left(\left(n_e^1 - 1\right) {p'}_e^1 + n_e^2 {p'}_e^2\right)\right| \\
&\quad \text{since } {n'}_e^1 = n_e^1 - 1, {n'}_e^2 = n_e^2 \\
&= \left|p_e^1 + \left(n_e^1 - 1\right)\left(p_e^1 - {p'}_e^1\right) + n_e^2\left(p_e^2 - {p'}_e^2\right)\right| \\
&= \left|p_e^1 + \left(n_e^1 - 1\right)\left(p_e^1 - {p'}_e^1\right) + n_e^2\left(\left(1 - p_e^1\right) - \left(1 - {p'}_e^1\right)\right)\right| \\
&= \left|p_e^1 + \left(n_e^1 - n_e^2 - 1\right)\left(p_e^1 - {p'}_e^1\right)\right|,
\end{aligned} \tag{9}$$

where p_e^1 is the probability of Class 1 is still the dominant class after adding noise. The last equality holds, because $p_e^2 = 1 - p_e^1$ and ${p'}_e^2 = 1 - {p'}_e^1$.

As for p_e^1, by Lemma 3,

$$\begin{aligned}
p_e^1 &= \Pr\left[n_e^1 + Z_1 \geq n_e^2 + Z_2\right] = \Pr\left[Z_2 - Z_1 \leq n_e^1 - n_e^2\right] \\
&= \begin{cases} 1 - \dfrac{e^{-\epsilon(n_e^1 - n_e^2)}}{2}\left(1 + \dfrac{\epsilon\left(n_e^1 - n_e^2\right)}{2}\right), & \text{if } n_e^1 - n_e^2 \geq 1, \\ \dfrac{e^{\epsilon(n_e^1 - n_e^2)}}{2}\left(1 - \dfrac{\epsilon\left(n_e^1 - n_e^2\right)}{2}\right), & \text{if } n_e^1 - n_e^2 \leq 0. \end{cases}
\end{aligned} \tag{10}$$

Since the probability p_e^1 (Eq. 10) takes different forms depending on whether $n_e^1 - n_e^2 \geq 0$ or not, we analyze the global sensitivity Δ_{gq} by two cases: Class 1 is the dominant class in the cell e for D, $n_e^1 - n_e^2 \geq 1$ or not, $n_e^1 - n_e^2 \leq 0$. We show that for $n_e^1 - n_e^2 \geq 1$, the upper bound of the global sensitivity is 1.1 and for $n_e^1 - n_e^2 \leq 0$, the upper bound is 0.5. Note that the two conditions cover all cases, since n_e^1 and n_e^2 are integers. Therefore, the global sensitivity can be bounded by 1.1.

Case 1: $\mathbf{n_e^1 - n_e^2 \geq 1}$.

In this case, ${n'}_e^1 - {n'}_e^2 = n_e^1 - 1 - n_e^2 \geq 0$. By Eqs. (9) and (10), we have

$$\begin{aligned}
\Delta_{\mathsf{gq}} = \Bigg| 1 &- \frac{e^{-\epsilon(n_e^1 - n_e^2)}}{2}\left(1 + \frac{\epsilon\left(n_e^1 - n_e^2\right)}{2}\right) + \\
&\left(n_e^1 - n_e^2 - 1\right)\left[\frac{e^{-\epsilon({n'}_e^1 - {n'}_e^2)}}{2}\left(1 + \frac{\epsilon\left({n'}_e^1 - {n'}_e^2\right)}{2}\right)\right. \\
&\left.- \frac{e^{-\epsilon(n_e^1 - n_e^2)}}{2}\left(1 + \frac{\epsilon\left(n_e^1 - n_e^2\right)}{2}\right)\right]\Bigg|.
\end{aligned}$$

By letting $x = n_e^1 - n_e^2$, we have

$$\begin{aligned}
\Delta_{\mathsf{gq}} &= \left|1 - \frac{e^{-\epsilon x}}{2} + (x - 1)\left[\frac{e^{-\epsilon(x-1)}}{2}\left(1 + \frac{\epsilon(x - 1)}{2}\right)\right.\right. \\
&\qquad \left.\left.\left(1 + \frac{\epsilon x}{2}\right) - \frac{e^{-\epsilon x}}{2}\left(1 + \frac{\epsilon x}{2}\right)\right]\right| \\
&= \left|1 + \frac{(x - 1)e^{-\epsilon(x-1)}}{2}\left(1 + \frac{\epsilon(x - 1)}{2}\right) - \frac{x e^{-\epsilon x}}{2}\left(1 + \frac{\epsilon x}{2}\right)\right|,
\end{aligned}$$

where $x \geq 1$.

For simplicity, let us consider the function $g_1(x) = \frac{x e^{-\epsilon x}}{2}(1 + \frac{\epsilon x}{2})$, and thus the sensitivity becomes $\Delta_{\mathsf{gq}} = |1 + g_1(x - 1) - g_1(x)|$.

Note that $g_1(x)$ is differentiable and its derivative $g_1'(x) = \frac{e^{-\epsilon x}}{4}(2 - \epsilon^2 x^2)$. Thus, by Lagrange's Mean Value Theorem, for any $x \geq 1$, there exists some ξ between $x - 1$ and x (thus $\xi > 0$), so that

$$\begin{aligned}
\Delta_{\mathsf{gq}} &= |1 + g_1(x - 1) - g_1(x)| \\
&= \left|1 - g_1'(\xi)\right| \\
&= \left|1 - \frac{e^{-\epsilon\xi}}{4}\left(2 - \epsilon^2\xi^2\right)\right|.
\end{aligned}$$

To bound the expression above, consider another function

$$h(x) = 1 - \frac{e^{-\epsilon x}}{4}\left(2 - \epsilon^2 x^2\right),$$

where $x > 0$. The function $h(x)$ reaches the maximum at the point $\frac{1+\sqrt{3}}{\epsilon}$ with the maximum value 1.1, increases in the interval $\left[0, \frac{1+\sqrt{3}}{\epsilon}\right]$ and decreases in the interval $\left(\frac{1+\sqrt{3}}{\epsilon}, \infty\right)$. When $x \in [0, \frac{1+\sqrt{3}}{\epsilon}]$, $h(0) \leq h(x) \leq h(\frac{1+\sqrt{3}}{\epsilon})$ which means $h(x) \in [0.5, 1.1]$. When $x \in \left(\frac{1+\sqrt{3}}{\epsilon}, \infty\right)$, $h(x)$ decreases and lies in $(1, 1.1]$, because $\lim_{x\to+\infty} h(x) = 1$. Therefore, in this case,

$$\Delta_{\mathsf{gq}} = |h(\xi)| \in [0.5, 1.1].$$

Case 2: $\mathbf{n_e^1 - n_e^2 \leq 0}$
In this case,

$$n_e^{'1} - n_e^{'2} = n_e^1 - 1 - n_e^2 < 0.$$

Similarly, by letting $x = n_e^1 - n_e^2$, we have

$$\begin{aligned}
\Delta_{\mathsf{gq}} &= \left| \frac{e^{\epsilon(n_e^1-n_e^2)}}{2}\left(1-\frac{\epsilon\left(n_e^1-n_e^2\right)}{2}\right) + \right. \\
&\quad \left(n_e^1-n_e^2-1\right)\left[\frac{e^{\epsilon(n_e^1-n_e^2)}}{2}\left(1-\frac{\epsilon\left(n_e^1-n_e^2\right)}{2}\right) - \right. \\
&\quad \left.\left. \frac{e^{\epsilon(n_e^1-n_e^2-1)}}{2}\left(1-\frac{\epsilon\left(n_e^1-n_e^2-1\right)}{2}\right)\right]\right| \\
&= \left|\frac{e^{\epsilon x}}{2}\left(1-\frac{\epsilon x}{2}\right)+(x-1)\right. \\
&\quad \left.\left[\frac{e^{\epsilon x}}{2}\left(1-\frac{\epsilon x}{2}\right)-\frac{e^{\epsilon(x-1)}}{2}\left(1-\frac{\epsilon(x-1)}{2}\right)\right]\right| \\
&= \left|\frac{xe^{\epsilon x}}{2}\left(1-\frac{\epsilon x}{2}\right)-\frac{(x-1)e^{\epsilon(x-1)}}{2}\left(1-\frac{\epsilon(x-1)}{2}\right)\right|,
\end{aligned}$$

where $x \leq 0$.

Similarly to Case 1, let $g_2(x) = \frac{xe^{\epsilon x}}{2}\left(1-\frac{\epsilon x}{2}\right)$, and then the sensitivity becomes $\Delta_{\mathsf{gq}} = |g_2(x) - g_2(x-1)|$.

The function $g_2(x)$ is differentiable and its first order derivative is $g_2'(x) = \frac{e^{\epsilon x}}{4}\left(2-\epsilon^2x^2\right)$. The derivative $g_2'(x)$ decreases when $x \in \left(-\infty, -\frac{1+\sqrt{3}}{\epsilon}\right)$, increases when $x \in \left[-\frac{1+\sqrt{3}}{\epsilon}, 0\right]$. And when $x = -\frac{1+\sqrt{3}}{\epsilon}$ the function $g_2'(x)$ reaches the minimum value -0.09. Thus, when $x \leq -\frac{1+\sqrt{3}}{\epsilon}$, $g_2'(x) \in [-0.09, 0)$ because $\lim_{x\to-\infty} g_2'(x) = 0$ and $g_2'(x) \in [-0.09, 0.5]$ when $x \in [-\frac{1+\sqrt{3}}{\epsilon}, 0]$. Applying Lagrange's Mean Value Theorem to $g_2(x)$, for any $x \leq 0$, there exists some η between $x-1$ and x, thus $\eta \leq 0$, so that

$$\begin{aligned}
\Delta_{\mathsf{gq}} &= |g_2(x) - g_2(x-1)| \\
&= \left|g_2'(\eta)\right| \\
&\leq 0.5.
\end{aligned}$$

In summary, by considering the above two cases, the global sensitivity for grid quality on binary classification Δ_{gq} is bounded by 1.1 and reaches its maximum when Case 1 holds, at

$$x^* = \frac{\epsilon e^\epsilon + \sqrt{2-(4-2e^\epsilon)\,e^\epsilon + \epsilon^2 e^\epsilon}}{-\epsilon + \epsilon e^\epsilon}.$$

The global maximum point x^* is obtained by taking derivative of $1 + g_1(x-1) - g_1(x)$. This completes the proof. □

Proof of Lemma 2

Proof Given a dataset D, without loss of generality we assume that the neighboring dataset of D is $D' = D - t$, where the tuple t is in cell e. The quality values of all cells other than cell e are the same. Denote l_t as the class label of t.

Since the approximation of the grid quality (Eq. 5) involves only the top two classes, and the quality values of all cells other than cell e are the same, the difference of the approximated grid quality function $\Delta_{\mathsf{gq}} = |\mathsf{gq}(D, g) - \mathsf{gq}(D', g)|$ is 0 or not depends on whether the count of Class l_t is in the top two class counts of cell e in D and D'. It is worth pointing out that the rank of Class l_t does not rise in $D' = D - t$ because deleting the tuple t can only decrease the count of Class l_t. Therefore, we bound Δ_{gq} by considering three separated cases: (1) Class l_t is not in the top two classes in D and D', (2) Class l_t is in the top two classes in D and D', and (3) Class l_t is in the top two classes in D but not in D'. For convenience, we use $l_t \in \{(1), (2)\}$ to represent the fact that the class count of lt is in top 2, and $l_t \notin \{(1), (2)\}$ to represent that it is not.

Case 1: $l_t \notin \{(1), (2)\}$ in both D and D'. Class l_t does not rank in top-2 in both D and D'. In this case, deleting the tuple t does not affect the first 2 classes. $n_e^{(1)} = n_e^{'(1)}, n_e^{(2)} = n_e^{'(2)}, p_e^{(1)} = p_e^{'(1)}$ and $p_e^{(2)} = p_e^{'(2)}$. So $\mathsf{gq}(D, g) = \mathsf{gq}(D', g)$, which means $\Delta_{\mathsf{gq}} = 0$.

Case 2: $l_t \in \{(1), (2)\}$ in both D and D'. Class l_t ranks in top-2 in both D and D'. In this case, the rank of l_t may change. So let us consider the following subcases.

Subcase 2.1: Class l_t ranks first (resp. second) in both D and D'. Then, we have $n_e^{(1)} = n_e^{'(1)} + 1$ and $n_e^{(2)} = n_e^{'(2)}$ (resp. $n_e^{(1)} = n_e^{'(1)}$ and $n_e^{(2)} = n_e^{'(2)} + 1$). Similar to the proof of Lemma 1, we obtain $\Delta_{\mathsf{gq}} \leq 1.1$.

Subcase 2.2: Class l_t ranks first in D and ranks second in D'. Deleting one tuple makes the class with the highest count become the second highest class, which occurs only when there is a tie between two highest classes in D. (This tie can be resolved the alphabetical order of class labels.) In this case, we have

$$n_e^{(1)} = n_e^{(2)}, n_e^{'(1)} = n_e^{(2)}, n_e^{'(2)} = n_e^{(1)} - 1. \tag{11}$$

Thus,

$$\begin{aligned}
\Delta_{\mathsf{gq}} &= |\mathsf{gq}(D, g) - \mathsf{gq}(D', g)| \\
&= \left|\left(n_e^{(1)}p_e^{(1)} + n_e^{(2)}p_e^{(2)}\right) - \left(n_e^{'(1)}p_e^{'(1)} + n_e^{'(2)}p_e^{'(2)}\right)\right| \\
&= \left|n_e^{(1)}\left(p_e^{(1)} + p_e^{(2)}\right) - \left(n_e^{(1)}p_e^{'(1)} + \left(n_e^{(1)} - 1\right)p_e^{'(2)}\right)\right| \\
&= \left|n_e^{(1)} - \left(n_e^{(1)} - p_e^{'(2)}\right)\right| \\
&= \left|p_e^{'(2)}\right| \leq 1,
\end{aligned}$$

where the third equality holds because of Eq. (11) and the fourth equality holds because $p_e^{(2)} = 1 - p_e^{(1)}$ and $p_e^{'(2)} = 1 - p_e^{'(1)}$.

Case 3: $l_t \in \{(1), (2)\}$ **in** D and $l_t \notin \{(1), (2)\}$ **in** D'. Class l_t ranks in top-2 in D, but does not rank in top-2 in D'. Similar to Subcase 2.2, this case occurs only when there is a tie among Class l_t and other $s(\geq 1)$ classes which are not ranked in top-2. Deleting the tuple t makes l_t ranked out of top-2. So, by our tie resolving rule, one of the s classes with the same count as Class l_t is ranked into top-2. For example, suppose the number of class $k = 3$, $n_e^1 = n_e^2 = n_e^3 = 10$, the first highest classes with counts $n_e^{(1)} = n_e^{(2)} = 10$. After removing the tuple t with label 1, we have $n_e'^2 = n_e'^3 = 10$, and $n_e'^1 = 9$. The class containing t is ranked out of top-2 and the class with label 3 is ranked into top-2, and the counts of the top two classes in D' are still the same, which means $n_e^{'(1)} = n_e^{'(2)} = 10$. That is to say,

$$n_e^{(1)} = n_e^{(2)}, n_e^{'(1)} = n_e^{(1)}, n_e^{'(2)} = n_e^{(1)}. \tag{12}$$

Similarly, we have

$$\begin{aligned}\Delta_{\mathsf{gq}} &= |\mathsf{gq}(D, g) - \mathsf{gq}(D', g)| \\ &= \left|\left(n_e^{(1)} p_e^{(1)} + n_e^{(2)} p_e^{(2)}\right) - \left(n_e^{'(1)} p_e^{'(1)} + n_e^{'(2)} p_e^{'(2)}\right)\right| \\ &= \left|n_e^{(1)}\left(p_e^{(1)} + p_e^{(2)}\right) - \left(n_e^{(1)} p_e^{'(1)} + n_e^{(1)} p_e^{'(2)}\right)\right| \\ &= \left|n_e^{(1)} - n_e^{(1)}\right| = 0,\end{aligned}$$

where the second equality holds because of Eq. (12).

In summary, the global sensitivity for the approximation of grid quality on multiclass classification $\Delta_{\mathsf{gq}} \leq 1.1$. □

References

1. Asuncion, A., Newman, D.: UCI machine learning repository (2010)
2. Bayardo, R.J., Agrawal, R.: Data privacy through optimal k-anonymization. In: ICDE, pp. 217–228 (2005)
3. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer, Secaucus (2006)
4. Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: the SuLQ framework. In: PODS, pp. 128–138 (2005)
5. Chang, C.C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**, 27:1–27:27 (2011)
6. Chaudhuri, K., Monteleoni, C.: Privacy-preserving logistic regression. In: NIPS, pp. 289–296 (2008)
7. Chaudhuri, K., Monteleoni, C., Sarwate, A.D.: Differentially private empirical risk minimization. J. Mach. Learn. Res. **12**, 1069–1109 (2011)
8. Cormode, G., Srivastava, D., Li, N., Li, T.: Minimizing minimality and maximizing utility: analyzing method-based attacks on anonymized data. PVLDB **3**(1–2), 1045–1056 (2010)
9. Devroye, L., Györfi, L., Lugosi, G.: A Probabilistic Theory of Pattern Recognition. Applications of Mathematics. Springer, Berlin (1996)
10. Dwork, C.: Differential privacy. In: ICALP, pp. 1–12 (2006)
11. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: TCC, pp. 265–284 (2006)
12. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: CCS, pp. 1322–1333 (2015)
13. Fredrikson, M., Lantz, E., Jha, S., Lin, S., Page, D., Ristenpart, T.: Privacy in pharmacogenetics: an end-to-end case study of personalized warfarin dosing. In: USENIX Security Symposium, pp. 17–32 (2014)
14. Friedman, A., Schuster, A.: Data mining with differential privacy. In: KDD, pp. 493–502 (2010)
15. Fung, B.C.M., Wang, K., Yu, P.S.: Top-down specialization for information and privacy preservation. In: ICDE, pp. 205–216 (2005)
16. Geng, X., Liu, T.Y., Qin, T., Li, H.: Feature selection for ranking. In: SIGIR, pp. 407–414 (2007)
17. Hay, M., Machanavajjhala, A., Miklau, G., Chen, Y., Zhang, D.: Principled evaluation of differentially private algorithms using dpbench. In: SIGMOD, pp. 139–154 (2016)
18. Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: KDD, pp. 279–288 (2002)
19. Jagannathan, G., Pillaipakkamnatt, K., Wright, R.N.: A practical differentially private random decision tree classifier. Trans. Data Priv. **5**, 273–295 (2012)
20. Kotz, S., Kozubowski, T., Podgorski, K.: The Laplace Distribution and Generalizations: A Revisit with Applications to Communications, Economics, Engineering, and Finance. Springer, Berlin (2001)
21. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Incognito: efficient full-domain k-anonymity. In: SIGMOD, pp. 49–60 (2005)
22. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: ICDE, p. 25 (2006)
23. Lei, J.: Differentially private m-estimators. In: NIPS, pp. 361–369 (2011)
24. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: FOCS, pp. 94–103 (2007)
25. Minnesota Population Center: Integrated Public Use Microdata Series, International: Version 6.5 [dataset]. University of Minnesota, Minneapolis (2017). https://doi.org/10.18128/D020.V6.5
26. Mohammed, N., Chen, R., Fung, B.C.M., Yu, P.S.: Differentially private data release for data mining. In: KDD, pp. 493–501 (2011)
27. Qardaji, W., Yang, W., Li, N.: Differentially private grids for geospatial data. In: ICDE, pp. 757–768 (2013)
28. Qardaji, W., Yang, W., Li, N.: Understanding hierarchical methods for differentially private histograms. PVLDB **6**(14), 1954–1965 (2013)
29. Ruggles, S., Genadek, K., Goeken, R., Grover, J., Sobek, M.: Integrated Public Use Microdata Series: Version 7.0 [dataset]. University of Minnesota, Minneapolis (2017). https://doi.org/10.18128/D010.V7.0
30. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22–26, 2017, pp. 3–18 (2017)
31. Su, D., Cao, J., Li, N., Bertino, E., Jin, H.: Differentially private k-means clustering. In: Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy, CODASPY 2016, New Orleans, LA, USA, March 9–11, 2016, pp. 26–37 (2016)
32. Su, D., Cao, J., Li, N., Bertino, E., Lyu, M., Jin, H.: Differentially private k-means clustering and a hybrid approach to private optimization. ACM Trans. Priv. Secur. **20**(4), 16:1–16:33 (2017)

33. Therneau, T.M., Atkinson, B.: Package: rpart (2014). http://cran.r-project.org/web/packages/rpart/rpart.pdf
34. Vinterbo, S A.: Differentially private projected histograms: construction and use for prediction. In: ECML PKDD'12, pp. 19–34 (2012)
35. Wong, R.C.W., Fu, A. W. C., Wang, K., Pei, J.: Minimality attack in privacy preserving data publishing. In: VLDB, pp. 543–554 (2007)
36. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: ICML, pp. 412–420 (1997)
37. Zhang, J., Cormode, G., Procopiuc, C.M., Srivastava, D., Xiao, X.: Privbayes: private data release via bayesian networks. In: SIGMOD '14, pp. 1423–1434 (2014)
38. Zhang, J., XiaoXia, X., Yang, Y., Zhang, Z., Winslett, M.: Privgene: differentially private model fitting using genetic algorithms. In: SIGMOD '13, pp. 665–676 (2013)
39. Zhang, J., Zhang, Z., Xiao, X., Yang, Y., Winslett, M.: Functional mechanism: regression analysis under differential privacy. PVLDB **5**(11), 1364–1375 (2012)