Data Min Knowl Disc (2017) 31:1544–1575
DOI 10.1007/s10618-017-0532-z

Differentially private nearest neighbor classification

Mehmet Emre Gursoy[1] · **Ali Inan**[2] · **Mehmet Ercan Nergiz**[3] · **Yucel Saygin**[4]

Received: 21 January 2017 / Accepted: 11 July 2017 / Published online: 21 July 2017

Abstract Instance-based learning, and the k-nearest neighbors algorithm (k-NN) in particular, provide simple yet effective classification algorithms for data mining. Classifiers are often executed on sensitive information such as medical or personal data. Differential privacy has recently emerged as the accepted standard for privacy protection in sensitive data. However, straightforward applications of differential privacy to k-NN classification yield rather inaccurate results. Motivated by this, we develop algorithms to increase the accuracy of private instance-based classification. We first describe the radius neighbors classifier (r-N) and show that its accuracy under differential privacy can be greatly improved by a non-trivial sensitivity analysis. Then, for k-NN classification, we build algorithms that convert k-NN classifiers to r-N

Responsible editors: Kurt Driessens, Dragi Kocev, Marko Robnik-Šikonja, Myra Spiliopoulou.

This research was funded by The Scientific and Technological Research Council of Turkey (TUBITAK) under Grant Number 114E261.

✉ Ali Inan
ainan@adanabtu.edu.tr

Mehmet Emre Gursoy
memregursoy@gatech.edu

Mehmet Ercan Nergiz
nergiz@gmail.com

Yucel Saygin
ysaygin@sabanciuniv.edu

1 College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA

2 Computer Engineering Department, Adana Science and Technology University, Adana, Turkey

3 Acadsoft Research, Gaziantep, Turkey

4 Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey

classifiers. We experimentally evaluate the accuracy of both classifiers using various datasets. Experiments show that our proposed classifiers significantly outperform baseline private classifiers (i.e., straightforward applications of differential privacy) and executing the classifiers on a dataset published using differential privacy. In addition, the accuracy of our proposed k-NN classifiers are at least comparable to, and in many cases better than, the other differentially private machine learning techniques.

Keywords Data mining · Differential privacy · k-Nearest neighbors

1 Introduction

Instance-based learning refers to algorithms that, instead of building explicit rules, generalizations or models, compare new instances with instances seen in training. They have several advantages over traditional learning methods, the greatest of which is their *lazy aspect* (Aggarwal 2014): Traditional methods create models that are myopic and not optimized towards specific test instances. Instance-based learning overcomes this by creating models that are specific to each test instance. Among instance-based learning algorithms, k-nearest neighbors (k-NN) is a simple yet powerful classification algorithm. It was designed with the assumption that neighboring (nearby) data points are likely to share the same label (class value). The k-NN algorithm has found applications in various contexts including biological and chemical data analysis (Horton and Nakai 1997), healthcare (Parry et al. 2010) and recommender systems. In Wu et al. (2008), it was identified as one of the top 10 most influential data mining algorithms.

k-NN must often be run on sensitive data such as medical records, user preferences or personal information. The privacy of sensitive data is of utmost importance. In recent years, *differential privacy* (ε-DP) has emerged as the prominent method in statistical data privacy. It states that the output of a computation should not overly depend on any individual record. A popular method to achieve ε-DP is to add random noise to an algorithm's outputs, where the noise is scaled according to the *sensitivity* of the algorithm.

In this paper, we propose algorithms for differentially private instance-based learning. The scenario we consider is illustrated in Fig. 1, and can be described as follows: A private dataset containing sensitive information resides with the data owner. This dataset contains *training instances* (i.e., observations) whose labels are known. The data analyst, an untrusted third party, has a collection of instances (called *test instances* here onwards) for which the labels are unknown. The data analyst poses his test instances to the data owner, the owner applies a differentially private instance-based learning algorithm to label each instance, and returns the results. The goal of our work is to make the learning algorithm accurate, while satisfying a certain level of privacy (ε). As we show in the next sections, the trivial baseline approaches yield low accuracy due to the amount of noise that must be added to satisfy ε-DP. Therefore we need more sophisticated algorithms that calculate the lowest amount of noise that both satisfies ε-DP and yields accurate classification results.

In our solution, we first show how to build accurate and private *radius neighbors* (r-N) classifiers. The r-N classifier implements a majority vote among neighbors within a

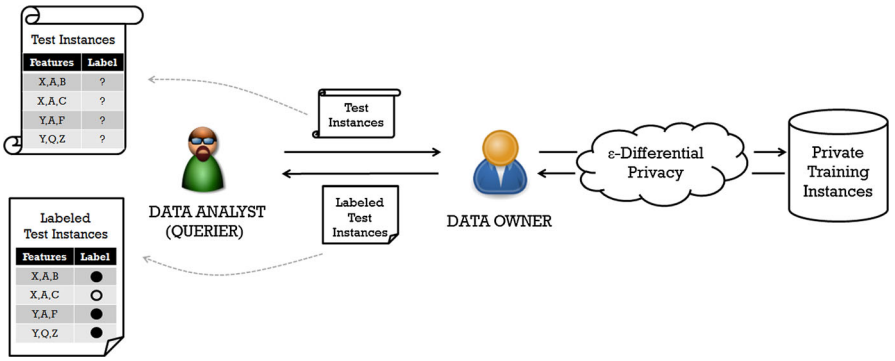

Fig. 1 Proposed scenario and privacy model

fixed given radius (name is due to scikit-learn 2017; Behley et al. 2015; Bentley 1975). To make r-N classifiers accurate, we perform *sensitivity analysis* on the proximity regions of test instances, i.e., we determine the potential overlaps between the radii r of test instances, and use this to calculate sensitivity. A novel graph structure, called the *region overlap graph*, is employed to make sensitivity analysis efficient and scalable. With the help of sensitivity analysis, we calculate the maximum potential effect of a record on the classification results. This allows us to find a tight upper bound on the amount of noise satisfying ε-DP.

Next, for k-NN classification, we convert from k-NN classification to r-N classification. In the non-private setting, this conversion is trivial: We can simply set r to be the distance between the test instance and its k'th nearest neighbor in the dataset. However, this clearly leaks the distances among the training instances, and consequently the data distribution. To remedy this problem, we propose two algorithms (one interactive, one non-interactive) that perform approximate conversions that satisfy ε-DP. We verbally and experimentally compare the two conversion algorithms, and argue that they are appropriate in different settings.

Although there exist many works on ε-DP machine learning (Ji et al. 2014; Sarwate and Chaudhuri 2013), to the best of our knowledge, ours is the first work on k-NN classification. We conjecture that this surprising lack of work is due to the difficulty in relating the notion of *sensitivity* in ε-DP with the k in k-NN. Note that in the worst case, removal of one training instance from the private dataset may cause the proximity region of a test instance to expand as much as to the boundaries of the data space. This makes it difficult to bound the sensitivity of k-NN classification. However, we naturally solve this problem by converting k-NN classifiers to r-N classifiers, where we can heuristically find a good r given a test instance and k. Experiments show that our proposed k-NN classifiers either outperform other ε-DP machine learning algorithms, or they have comparable accuracy.

The rest of this paper is organized as follows: In Sect. 2, we give the preliminary definitions concerning our data model, ε-DP and graph terminology. In Sect. 3, we describe the baseline approach as well as our proposed approach for private r-N classification. In Sect. 4, we describe baseline and proposed approaches for private

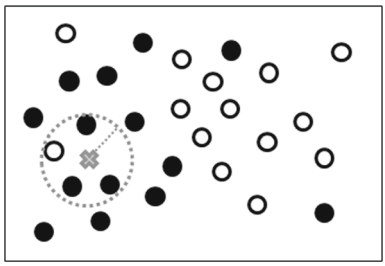

Fig. 2 Sample r-N classifier. *Black* and *white dots* are the training instances, and their color is their label. The *grey* **x** denotes the test instance, and the *dashed circle* is its proximity region

k-NN classification. We present our experimental evaluation in Sect. 5 and related work in Sect. 6. In Sect. 7, we discuss some of the shortcomings of our work and propose future research directions. We conclude in Sect. 8.

2 Preliminaries

Let D be a tabular dataset containing d attributes. The domain of each attribute A_i is denoted $\Omega(A_i)$. One of the d attributes is designated as the *label* or *class* attribute, denoted L. The remaining set $\mathcal{F}$ of attributes are called *features*. We assume that the label is categorical, whereas the features are numeric (discrete or continuous). Each row in the private dataset D constitutes a training instance. For any instance $t \in D$, let $t[L]$ denote the value of the label of t. We treat each instance as a (d-1)-dimensional point in the data space. The data space is determined by the cross product $\Omega(D) = \prod_{A_i \in \mathcal{F}} \Omega(A_i)$. We give a sample visualization in Fig. 2 where $d = 3$.

2.1 Differential privacy

Differential privacy ensures that the result of an algorithm is not overly dependent on any instance. It states that there should be a strong probability of producing the same output even if an instance was added to or removed from the dataset.

Definition 1 (*Neighboring datasets*) Two datasets D, D' are called neighboring datasets, denoted by $|D \Delta D'| = 1$, if one can be obtained from the other by either adding or removing one instance, i.e., $|(D - D') \cup (D' - D)| = 1$.

Definition 2 (ε-*differential privacy*) A randomized algorithm $\mathcal{A}$ is ε-differentially private (ε-DP) if for all neighboring datasets D, D' and for all subsets of possible outcomes of the algorithm $S \subseteq Range(\mathcal{A})$,

$$Pr[\mathcal{A}(D) \in S] \leq e^{\varepsilon} \times Pr[\mathcal{A}(D') \in S]$$

where the probabilities are over the randomness of $\mathcal{A}$.

Here, ε is the privacy parameter that controls how much an advesary can distinguish between D and D'. Smaller ε yields higher privacy.

There are well-known, general-purpose building blocks that help in designing differentially private algorithms. Next, we present these building blocks. We start with the definition of *sensitivity*.

Definition 3 (*Sensitivity*) Let $f : D \rightarrow \mathbb{R}^m$ be a function that maps a dataset D into a fixed-size vector of m real numbers. The sensitivity of f is defined as:

$$\Delta f := \max_{D,D'} ||f(D) - f(D')||$$

for all $|D \Delta D'| = 1$, where $||.||$ denotes the L_1 norm.

A popular ε-DP algorithm for answering numeric queries is the Laplace Mechanism, defined formally as follows. In words, when answering a set of numeric queries (e.g., a set of count queries) the Laplace Mechanism retrieves the true answers of these queries, and then perturbs each answer by adding random noise scaled according to their sensitivity.

Definition 4 (*Laplace mechanism*) Let $Lap(\sigma)$ denote a random variable sampled from the Laplace distribution with mean 0 and scale parameter σ. For a numeric-valued function $f : D \rightarrow \mathbb{R}^s$, the algorithm $\mathcal{A}$ that answers f by:

$$\mathcal{A}(f, D) = f(D) + < Y_1, \ldots, Y_s >$$

is ε-DP if Y_i are i.i.d. random variables drawn from $Lap(\sigma)$ where $\sigma \geq \Delta f/\varepsilon$ (Dwork 2008).

We now consider the exponential mechanism for answering categorical queries (McSherry and Talwar 2007). The exponential mechanism is useful for selecting a discrete output r from a domain R in a differentially private manner. It employs a utility function (i.e., quality criterion) q that associates each output $r \in R$ with a non-zero probability of being selected. The sensitivity of the quality criterion q is defined as:

$$\Delta q := \max_{\forall r \in R,\ |D \Delta D'|=1} ||q(D, r) - q(D', r)||$$

Definition 5 (*Exponential mechanism*) Let $q : (D \times R) \rightarrow \mathbb{R}$ be a utility function for a dataset D and a domain of discrete outputs R. The algorithm $\mathcal{A}$ that returns the output $r \in R$ with probability proportional to $e^{\frac{\varepsilon q(D,r)}{2\Delta q}}$ satisfies ε-DP (McSherry and Talwar 2007).

Differentially private algorithms have the following composition properties, which allow us to build more complex algorithms by combining the aforementioned building blocks (McSherry 2009).

Definition 6 (*Composition properties*) Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be ε_1 and ε_2-DP algorithms, respectively. Then, we have:

– Sequential Composition: Releasing the output of $\mathcal{A}_1(D)$ and $\mathcal{A}_2(D)$ satisfies $(\varepsilon_1 + \varepsilon_2)$-DP.

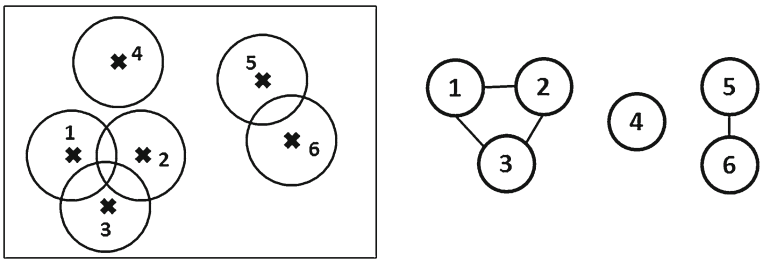

Fig. 3 Conversion from test instances in $\Omega(D)$ (on the *left*) to a region overlap graph (on the *right*)

- Parallel Composition: For $D_1 \cap D_2 = \emptyset$, releasing $\mathcal{A}_1(D_1)$ and $\mathcal{A}_2(D_2)$ satisfies $max(\varepsilon_1, \varepsilon_2)$-DP.
- Immunity to Post-Processing: Post processing the output of $\mathcal{A}_1(D)$ without accessing D, and releasing the processed version satisfies ε_1-DP.

Due to these properties, ε is also called the *privacy budget*. Given a total budget ε, the goal is to create an ε-DP algorithm $\mathcal{A}$ by cleverly combining the building blocks according to the composition properties. The goodness of $\mathcal{A}$ is measured by its accuracy. In our case, since $\mathcal{A}$ is a classification algorithm, its goodness is measured by the classification accuracy.

2.2 Graph terminology

Since our solution involves graph modeling of classification queries, in this section we give an introduction to graph terminology and the maximum clique problem. Let $G(V, E)$ be an undirected graph with vertex set V and edge set $E \subseteq V \times V$. A *clique* C of G is a subset of V such that every two vertices in C are adjacent, i.e., $\forall u, v \in C, (u, v) \in E$. A clique is a *maximum clique* if its cardinality is the largest among all the cliques of the graph. A graph may contain multiple maximum cliques.

Definition 7 (*Maximum clique problem*) Given a graph $G(V, E)$, the maximum clique problem is to find a clique C of G that has the highest cardinality. We denote the cardinality of C, often called the clique number of G, by $MCS(G)$.

For example, in Fig. 3, vertices $\{5, 6\}$ form a clique but this clique is not the maximum clique. Vertices $\{1, 2, 3\}$ form the maximum clique. For this graph, $MCS(G) = 3$.

The maximum clique problem (MCP) has a wide range of applications, and is among the most studied combinatorial problems. Even though MCP is NP-complete (Karp 1972), due to its practical relevance, there has been significant effort for finding efficient solutions. We refer the interested reader to Wu and Hao (2015) for a recent survey on solving the MCP.

3 Private r-N classifiers

We start by introducing r-N classifiers on non-private data, and then describe our ε-DP algorithms for r-N classifiers on private data. Recall that D is the dataset containing training instances, and we treat the instances in D as multi-dimensional points in $\Omega(D)$. Let x denote a test instance for which $x[L]$ is unknown, and r denote a proximity parameter (i.e., radius). Here onwards we use the combination (x, r) to refer to a test instance.

Definition 8 (*Counting query*) Let $\delta(\cdot, \cdot)$ be a distance function between two instances (i.e., points). The counting query $c^D_{x,r}(I)$ returns the number of instances $t \in D$ such that $\delta(x, t) \leq r$ and $t[L] = I$.

We remove the superscript D from c^D when D is clear in the context. We require the distance function $\delta(\cdot, \cdot)$ to be a *metric*. Appropriate functions include Euclidean distance, Manhattan distance etc. (Doherty et al. 2007). Without loss of generality, we use Euclidean distance in this work.

Definition 9 (*Radius neighbors classifier*) Let $\mathcal{L}$ be the set of all possible labels, and the counting query c be defined as above. Then, given a test instance (x, r), the r-N classifier assigns the following label to x:

$$x[L] = \underset{I \in \mathcal{L}}{\operatorname{argmax}}\, c_{x,r}(I)$$

An example is given in Fig. 2. We plot the test instance (x, r) among the training instances. In two dimensions and using Euclidean distance, the proximity region of (x, r) boils down to a circle with center x and radius r. Then, the r-N classifier performs a majority vote among the instances within the circle. In this case, there are 3 black and 1 white instances in the proximity region of x, thus the r-N classifier assigns $x[L] = \mathit{black}$. In case of ties, e.g., 2 black and 2 white instances within the circle, we can randomly select one label.

3.1 First-cut solutions

We can implement the r-N classifier privately using either the exponential mechanism or the Laplace mechanism. The exponential mechanism chooses an answer probabilistically while favoring the best candidate $I \in \mathcal{L}$. The Laplace mechanism obtains noisy counts nc_I by adding random Laplace noise to the actual counts $c(I)$, and then deterministically chooses the best candidate based on the noisy counts. In this case, the two mechanisms perform comparably and can be used interchangeably (Machanavajjhala et al. 2011). We give the implementation with the Laplace mechanism in Algorithm 1 and the implementation with the exponential mechanism in Algorithm 2. The Laplace implementation will be useful also in the next sections.

Algorithm 1 Private r-N classifier using the Laplace mechanism

Input: Private training dataset D, set of labels $\mathcal{L}$, test instance (x, r), privacy budget ε'
Output: $x[L]$: the label of x
1: **for** $I \in \mathcal{L}$ **do**
2: Let $nc_I \leftarrow c_{x,r}(I) + Lap(1/\varepsilon')$
3: **end for**
4: **return** $\text{argmax}_{I \in \mathcal{L}}\, nc_I$

Theorem 1 *Algorithm 1 satisfies ε'-DP.*

Proof Since $c_{x,r}$ is a count query, $\Delta c = 1$. On line 2, actual counts are perturbed using the Laplace mechanism. Every instance $t \in D$ has a single label, and thus each $c_{x,r}(I)$ for $I \in \mathcal{L}$ is executed on a disjoint subset of D. It follows from the Laplace mechanism and the parallel composition property that Algorithm 1 satisfies ε'-DP. □

Algorithm 2 Private r-N classifier using the exponential mechanism

Input: Private training dataset D, set of labels $\mathcal{L}$, test instance (x, r), privacy budget ε'
Output: $x[L]$: the label of x
1: **return** I with probability $\propto e^{\frac{\varepsilon' c_{x,r}(I)}{2}}$, for all $I \in \mathcal{L}$

The proof that Algorithm 2 satisfies ε'-DP follows trivially from the exponential mechanism with $R = \mathcal{L}$, utility function $q = c_{x,r}$, and $\Delta c = 1$.

The baseline classifier Now, we study the problem of classifying multiple test instances with different proximity parameters. Let X denote a list of test instances $X = ((x_1, r_1), \ldots, (x_{|X|}, r_{|X|}))$. We wish to obtain $O = (L_1, \ldots, L_{|X|})$ where $L_i = x_i[L]$ as predicted by an r-N classifier. The non-private r-N classifier simply uses Definition 9 on each (x_i, r_i) independently.

Following this intuition, the baseline ε-DP r-N classifier is designed as follows: Given X, the private training dataset D and privacy budget ε, the baseline classifier invokes Algorithm 1 (or 2) for each $(x_i, r_i) \in X$ with $\varepsilon' = \frac{\varepsilon}{|X|}$. That is, each test instance is treated independently with an $\frac{\varepsilon}{|X|}$ share of the privacy budget. The baseline classifier satisfies ε-DP due to the sequential composition property.

As we will show in Sect. 5, the baseline classifier suffers from the over-division of the privacy budget. With a budget of $\varepsilon = 1.0$ and a reasonable number of test instances (e.g., $|X| = 100$), each test instance receives only $\varepsilon' = 0.01$. Consider the following example: For a binary class label (a/b) we have a test instance (x_i, r_i) for which 80 training instances in x's proximity vote a and 20 vote b. We expect the r-N classifier to clearly prefer a. However, if we follow the calculations in Algorithm 2, we obtain that the baseline classifier returns a with probability $\frac{e^{\frac{0.01\times 80}{2}}}{e^{\frac{0.01\times 80}{2}} + e^{\frac{0.01\times 20}{2}}} \approx 57\%$, which is only marginally better than a random guess. If we instead had a larger ε', we would

have obtained a significant preference towards a. This motivates our solution in the next section. [1]

3.2 Proposed solution

The baseline classifier suffers from a poor allocation of the privacy budget since it classifies each test instance sequentially and independently. Our proposed solution finds a better strategy using *sensitivity analysis* on the *collection* of test instances instead of treating them independently. This requires a careful analysis of the proximity regions of (x_i, r_i) to understand: (i) sets of test instances that can be labeled in parallel, and (ii) the combined sensitivity of test instances that must be labelled sequentially.

Algorithm 3 Proposed private r-N classifier

Input: Training dataset D, set of labels $\mathcal{L}$, list of test instances X, total privacy budget ε
Output: List of predicted labels $(L_1, \ldots, L_{|X|})$
1: Build region overlap graph $\mathcal{G}$ of X
2: **for** each disconnected subgraph $\mathcal{G}_{sub}$ in $\mathcal{G}$ **do**
3: Let $\varepsilon' \leftarrow \varepsilon / MCS(\mathcal{G}_{sub})$
4: **for** $(x_i, r_i) \in \mathcal{G}_{sub}$ **do**
5: $L_i \leftarrow$ Invoke Algorithm 1 with D, $\mathcal{L}$, (x_i, r_i), ε'
6: **end for**
7: **end for**
8: **return** $(L_1, \ldots, L_{|X|})$

The overview of our solution is given in Algorithm 3. The solution can be described in four steps: (i) First, we build a *region overlap graph* $\mathcal{G}$. (ii) We find the disconnected subgraphs $\mathcal{G}_{sub}$ in $\mathcal{G}$, and assign a privacy budget of ε to each $\mathcal{G}_{sub}$. This is because disconnected subgraphs can be executed via parallel composition. (iii) We show that the sensitivity of a subgraph $\mathcal{G}_{sub}$ is equal to its maximum clique size, $MCS(\mathcal{G}_{sub})$. (iv) Finally, we invoke the ε'-DP r-N classifier that uses the Laplace mechanism with $\varepsilon' = \varepsilon / MCS(\mathcal{G}_{sub})$ to obtain labels.

Building a region overlap graph $\mathcal{G}$ We define the region overlap graph below. Its purpose is to provide a compact summary of the intersections between the proximity regions of test instances. When the data dimensionality is high or there are many test instances, searching through $\Omega(D)$ to find all intersections is unfeasible. The region overlap graph enables us to perform efficient intersection search using well-known graph and clique algorithms.

[1] Note that Algorithm 2 satisfies ε'-DP even if there is only one training instance within the specified radius, since it always returns an answer probabilistically. We illustrate with the following example: For the binary classification task (a/b), and let there be 1 training instance within the radius of the specified test instance, with label a. Let $\varepsilon = 1.0$. Then, the score of a is: $\varepsilon^{1 \times \frac{1}{2}} = 1.65$, and the score of b is $\varepsilon^{1 \times \frac{0}{2}} = 1$. Thus, Algorithm 2 returns a with probability $\frac{1.65}{1+1.65}$ and b with probability $\frac{1}{1+1.65}$. Hence, there is a significant probability of returning b despite the only training instance within the radius has label a.

Definition 10 (*Region overlap graph*) Let $X = ((x_1, r_1), \ldots, (x_{|X|}, r_{|X|}))$ be the list of test instances. The region overlap graph $\mathcal{G}$ of X is an undirected and unweighted graph with vertex set V and edge set E, built as follows:

- Each $(x_i, r_i) \in X$ is represented by a vertex $V_i \in V$.
- There exists an edge $e \in E$ between V_i and V_j if and only if the proximity regions of (x_i, r_i) and (x_j, r_j) intersect, i.e., $\delta(x_i, x_j) \leq r_i + r_j$.

An example conversion is given in Fig. 3. The regions of x_1, x_2 and x_3 intersect, and therefore their vertices are adjacent in $\mathcal{G}$. x_5 and x_6's regions intersect, and therefore their vertices are adjacent. x_4's region does not intersect with any other instances, and therefore its vertex in $\mathcal{G}$ is disconnected from every other vertex.

Finding disconnected subgraphs in $\mathcal{G}$ We say that two subgraphs $\mathcal{G}^1_{sub}$ and $\mathcal{G}^2_{sub}$ are disconnected if there exists no path between any vertex $v^1 \in \mathcal{G}^1_{sub}$ and $v^2 \in \mathcal{G}^2_{sub}$. Given the full region overlap graph $\mathcal{G}$, we divide $\mathcal{G}$ into its disconnected subgraphs $\mathcal{G}^1_{sub}, \ldots, \mathcal{G}^n_{sub}$ using depth first search. Then, given the total privacy budget ε for $\mathcal{G}$, we assign ε to each $\mathcal{G}_{sub}$, since there can be no intersections between the proximity regions of the instances in different $\mathcal{G}^i_{sub}$. The following lemma proves the privacy guarantees of this procedure.

Lemma 1 *Assigning a budget of* ε *per disconnected* $\mathcal{G}^i_{sub}$ *in* $\mathcal{G}$ *satisfies* ε*-DP.*

Proof Let $\mathcal{G}^1_{sub}$ and $\mathcal{G}^2_{sub}$ be two disconnected subgraphs in $\mathcal{G}$. By definition of disconnected subgraphs, we know that there does not exist an edge in $\mathcal{G}$ between any $v_1 \in \mathcal{G}^1_{sub}$ and any $v_2 \in \mathcal{G}^2_{sub}$. By construction of $\mathcal{G}$ (i.e., Definition 10) this implies that there are no intersections between any pair of $(x_1, r_1) \in \mathcal{G}^1_{sub}$ and $(x_2, r_2) \in \mathcal{G}^2_{sub}$. Thus, the test instances in $\mathcal{G}^1_{sub}$ and $\mathcal{G}^2_{sub}$ concern disjoint subsets of D. This satisfies the parallel composition property in Definition 6. Following this and assuming a total budget of ε, assigning ε to $\mathcal{G}^1_{sub}$ and ε to $\mathcal{G}^2_{sub}$ satisfies $max(\varepsilon, \varepsilon) = \varepsilon$-DP. □

We exemplify this lemma using Fig. 3. The three disconnected subgraphs are: $\mathcal{G}^1_{sub} = \{x_1, x_2, x_3\}$, $\mathcal{G}^2_{sub} = \{x_4\}$ and $\mathcal{G}^3_{sub} = \{x_5, x_6\}$. As can be seen from the spatial distribution of the test instances (on the left), the three disconnected subgraphs concern disjoint subsets of the data space $\Omega(D)$.

Finding the sensitivity of a disconnected subgraph $\mathcal{G}_{sub}$ According to Definition 3, sensitivity measures the maximum possible change in the answers of a function in case it was executed on a neighboring dataset (i.e., in case a training instance was added to or removed from D). Without sensitivity analysis, the baseline classifier assumes that one training instance can affect the classification of all test instances in $\mathcal{G}_{sub}$. However, this is often not true: Consider the example in Fig. 4. For the test instances on the left, we cannot add a new training instance that would simultaneously affect all $\{x_1, x_2, x_3\}$, since they do not have a common intersection. On the other hand, for the test instances on the right, adding a new training instance to the common intersection of $\{x_1, x_2, x_3\}$ would affect all three. From this, we observe that the intersections *within* each $\mathcal{G}_{sub}$ are also important for sensitivity calculation. We also observe that the graphs of the two cases are different—the graph on the left has $MCS = 2$ but the one on the right has $MCS = 3$.

Fig. 4 Intersections within a subgraph determine its sensitivity

Lemma 2 *For a set of test instances $X' \subseteq X$ such that $|X'| > 1$, if the instances' proximity regions have a non-empty common intersection, then the vertices V representing X' form a clique of $\mathcal{G}$.*

Proof Let $X' = ((x_1, r_1), \ldots, (x_n, r_n))$ be the set of test instances with non-empty common intersection. This implies that $\forall$ pairs $(x_i, r_i), (x_j, r_j) \in X'$, the proximity regions of (x_i, r_i) and (x_j, r_j) intersect. In other words, all pairs of instances in X' pairwise intersect. By Definition 10, this implies that the representation of X' in $\mathcal{G}$, i.e., $V' = (v_1, \ldots, v_n) \in \mathcal{G}$, there exists an edge between (v_i, v_j) for all pairs $v_i, v_j \in V'$. By the definition of cliques, V' formulates a clique of $\mathcal{G}$. □

Lemma 3 *$MCS(\mathcal{G}_{sub})$ is an upper bound on the number of test instances in $\mathcal{G}_{sub}$ that can be affected by the addition or removal of a training instance.*

Proof We prove by contradiction: Assume that the number of test instances in $\mathcal{G}_{sub}$ that can be affected by one training instance is $\mathcal{G}_{sub} + m$, where m is a positive integer. Then, $MCS(\mathcal{G}_{sub}) + m$ test instances must have a common intersection to which we add/remove an instance so that we affect all of them simultaneously. However, a common intersection with $MCS(\mathcal{G}_{sub}) + m$ test instances implies a clique of size $MCS(\mathcal{G}_{sub}) + m$ due to Lemma 2. This clique is larger than $MCS(\mathcal{G}_{sub})$, which, by definition of MCS yields a contradiction. □

Recall that sensitivity is concerned with the *maximum* possible change by adding or removing an instance. We observe that in order to cause the maximum change, the instance should be added to/removed from the *maximum* common intersection, i.e., the region in $\Omega(D)$ where the highest number of test instances intersect. According to Lemma 3, $MCS(\mathcal{G}_{sub})$ is an upper bound on the maximum common intersection. Thus, it is also an upper bound for the sensitivity of $\mathcal{G}_{sub}$.

We prove the privacy guarantee of Algorithm 3 formally in Theorem 2. But first, we note the convenience of using the region overlap graph for finding sensitivity: It allows us to bound sensitivity simply by finding the maximum clique size. As benchmarked in Wu and Hao (2015), state of the art algorithms for finding a maximum clique take less than 5 s even for large graphs (e.g., 400–600 vertices). This makes our approach very efficient.

Theorem 2 *Algorithm 3 satisfies ε-DP.*

Proof In Lemma 1 we showed that treating each disconnected subgraph $\mathcal{G}_{sub}$ individually and assigning budget ε satisfies ε-DP. Thus it suffices here to show that the intra-$\mathcal{G}_{sub}$ allocation of the privacy budget (between lines 3–6) satisfies ε-DP.

We write the data accesses made by Algorithm 3 as a workload $\boldsymbol{W}$. Let $\mathcal{G}_{sub}$ be the subgraph of interest, and $X = ((x_1, r_1), \ldots, (x_n, r_n))$ denote the test instances represented by $\mathcal{G}_{sub}$. Then, Algorithm 3 calls Algorithm 1 (on line 5) for each $(x_i, r_i) \in X$. Within each call, Algorithm 1 executes a count query $c_{x_i,r_i}(I)$ for each $I \in \mathcal{L}$. This yields the following workload:

$$\begin{aligned}\boldsymbol{W} &= \bigcup_{(x_i,r_i)\in X} \left(\bigcup_{I\in\mathcal{L}} c_{x_i,r_i}(I) \right) \\ &= \bigcup_{I\in\mathcal{L}} \left(\bigcup_{(x_i,r_i)\in X} c_{x_i,r_i}(I) \right) = \bigcup_{I\in\mathcal{L}} f^I\end{aligned}$$

denoting $f^I := (c_{x_1,r_1}(I), \ldots, c_{x_n,r_n}(I))$. By Theorem 1 and the fact that each training instance has exactly one label, f^I concern disjoint subsets of the data. Therefore $\forall I \in \mathcal{L}$ functions f^I can be executed using parallel composition, receiving budget ε'. It remains to show Algorithm 3's selection of $\varepsilon' = \varepsilon / MCS(\mathcal{G}_{sub})$ yields ε-DP.

The true answers of f^I are perturbed with noise: $Lap(1/\varepsilon') = Lap(MCS(\mathcal{G}_{sub})/\varepsilon)$. By Lemma 3 we have $MCS(\mathcal{G}_{sub}) \geq \Delta f^I$, and therefore $MCS(\mathcal{G}_{sub})/\varepsilon \geq \Delta f^I/\varepsilon$. Hence we conclude by Definition 4 that Algorithm 3 satisfies ε-DP. □

4 Private k-NN classifiers

We now shift our direction towards our main goal, which is private k-NN classifiers. First, we give a formal definition of non-private k-NN classifiers. Let D be the dataset containing training instances, x be the test instance, k be the parameter in k-NN (i.e., number of nearest neighbors considered by the classifier), and $\delta(\cdot, \cdot)$ be a distance function. Similar to the previous section, we use the combination (x, k) to refer to a test instance, and drop the superscript D in the counting query when D is clear in the context.

Definition 11 (*k-NN counting query*) Let $D_{x,k} \subseteq D$ denote the collection of k instances in D nearest to x according to a distance function δ. The counting query $q^D_{x,k}(I)$ returns the number of instances $t \in D_{x,k}$ such that $t[L] = I$.

Definition 12 (*k-NN classifier*) Let $\mathcal{L}$ be the set of all possible labels, and the k-NN counting query q be defined as above. Then, given a test instance (x, k), the k-NN classifier assigns the following label to x:

$$x[L] = \operatorname*{argmax}_{I\in\mathcal{L}} q_{x,k}(I)$$

4.1 First-cut solution

Given a test instance (x, k), the k-NN classifier retrieves the k closest instances to x an performs a majority vote among them. Similar to r-N, this procedure can be

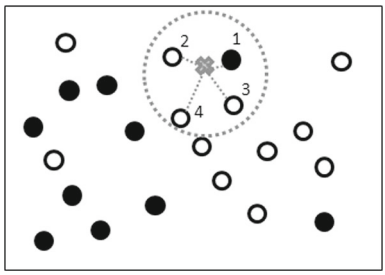

Fig. 5 Sample k-NN classifier, with $k = 4$. It can be trivially converted to an r-N classifier, where $r = \delta(x, x^4)$

implemented ε-DP using the Laplace or the exponential mechanism. A naive first-cut implementation with the exponential mechanism is given in Algorithm 4.

Algorithm 4 Private k-NN classifier using the exponential mechanism

Input: Private training dataset D, set of labels $\mathcal{L}$, test instance (x, k), privacy budget ε'
Output: $x[L]$: the label of x

1: **return** I with probability $\propto e^{\frac{\varepsilon' q_{x,k}(I)}{2}}$, for all $I \in \mathcal{L}$

The baseline classifier For the non-private k-NN classifier, the straightforward method to classify a list of test instances $X = ((x_1, k_1), \ldots, (x_{|X|}, k_{|X|}))$ is by independently running the k-NN classifier on each instance.

The baseline ε-DP k-NN classifier is designed as follows: Given X, the private training dataset D and privacy budget ε, the baseline classifier invokes Algorithm 4 for each $(x_i, k_i) \in X$ with $\varepsilon' = \frac{\varepsilon}{|X|}$. The output of Algorithm 4 is assigned as the label of x_i, i.e., $x_i[L]$. The baseline k-NN classifier is similar to the baseline r-N classifier, and thus suffers from the over-division of the privacy budget.

4.2 Proposed solution

Our proposed solution converts a k-NN classifier with (x, k) to an r-N classifier with (x, r). In the non-private setting, this conversion is trivial, using the following procedure: (1) Find the kth nearest neighbor of x in D. Let x^k denote this instance. (2) Set $r = \delta(x, x^k)$ and run the r-N classifier with (x, r). An example is given in Fig. 5. Given a test instance x and $k = 4$, we find the 4th nearest instance to x and set r according to the distance between x and the 4th nearest instance. Then, the r-N classifier can run seamlessly.

In the non-private setting, the conversion can be done with perfect accuracy using the procedure above. That is, the converted r-N classifier would behave the same as the original k-NN classifier. (For simplicity, we ignore cases where there might be other training instances $x^{k'}$ with distance $\delta(x, x^{k'}) = \delta(x, x^k)$.) However, perfect conversion is not possible in the private setting since we need to access the private data D to learn the appropriate r, and access is only granted through an ε-DP interface.

Hence, during the conversion process we will have to introduce some randomness or noise to satisfy ε-DP.

Then, given a test instance (x, k) our task becomes to find an accurate r differentially privately, for which the k-NN classifier with (x, k) would behave similarly to the r-N classifier with (x, r). Consider the trivial approach of adding Laplace noise: We retrieve the actual r and add Laplace noise to it. This approach is greatly inaccurate because in practice, we expect r to be very small compared to the amount of Laplace noise we are adding. For example, consider several thousand training instances distributed uniformly in $\Omega(D)$, which is a unit square of size 1. Assuming we have a reasonable k, e.g., $k = 50$, the actual r would be small, e.g., $r = 0.01$. The variance of the Laplace noise, however, would be proportional to the size of $\Omega(D)$, and a random sample from this noise could easily be significantly larger than 1. Adding this to $r = 0.01$ would destroy its accuracy, and lead to results that are far too inaccurate.

As a result, we need to spend some effort in designing algorithms that find r accurately. In the next section, we describe two algorithms (one interactive and one non-interactive) to accomplish this task. For now, let us assume that conversion is done using either of the two algorithms. Then, our proposed solution for ε-DP k-NN classification is given in Algorithm 5. The idea is to convert each k-NN instance to an r-N instance with some portion of the privacy budget, and then spend the rest of the budget to classify the converted r-N instances. The division of the budget is controlled by the weight parameter, w. In our experiments we use $w = 0.5$ for an equal budget share. Algorithm 5 satisfies ε-DP due to the sequential composition property, as long as our conversion algorithms (Algorithm 6 and 7) satisfy $\hat{\varepsilon}$-DP, where $\hat{\varepsilon} = w \cdot \varepsilon$.

Algorithm 5 Proposed private k-NN classifier

Input: Private training dataset D, set of labels $\mathcal{L}$, list of k-NN test instances $X = ((x_1, k_1), \ldots, (x_{|X|}, k_{|X|}))$, total privacy budget ε, weight parameter w

Output: List of predicted labels $(L_1, \ldots, L_{|X|})$

1: Invoke Algorithm 6 or 7 with D, $\mathcal{L}$, X and $\hat{\varepsilon} = w \cdot \varepsilon$ to obtain X', the list of corresponding r-N test instances

2: **return** Invoke Algorithm 3 with D, $\mathcal{L}$, X' and $(1 - w) \cdot \varepsilon$

At this point, we reiterate the advantages of converting k-NN to r-N: (1) As stated in Sect. 1, the worst case sensitivity of k-NN classification is hard to bound, and can be as large as the data space. This is because the removal of a record may cause the k'th nearest neighbor of a test instance to be arbitrarily far: In the worst case the new distance could be equal to the largest distance in $\Omega(D)$, which implies large sensitivity (and high amounts of noise) compared to typical values of r. However, via conversion to an r-N classifier, we can approximate and fix r, circumventing this problem. (2) In Sect. 3.2, we proposed a solution that increases the classification accuracy of r-N classifiers using sensitivity analysis. Conversion from k-NN to r-N means k-NN classifiers can also benefit from the same sensitivity analysis and its accuracy benefits.

4.3 Conversion from k-NN To r-N

Interactive (per-instance) algorithm The pseudocode for the algorithm is given in Algorithm 6. Recall that for a dataset D, $\mathcal{F}$ denotes its set of features. Given a list of k-NN test instances X, the algorithm proceeds by iterating over the list and converting each instance in X to an r-N test instance, by performing the following. First, on line 3 it finds r_{UNIF}, the most suitable radius if the data D was uniformly distributed in $\Omega(D)$. For this, we note that a hypersphere in $|\mathcal{F}|$ dimensions with radius r_{UNIF} has volume equal to:

$$\frac{\pi^{|\mathcal{F}|/2}}{\Gamma\left(\frac{|\mathcal{F}|}{2}+1\right)} \cdot r_{\text{UNIF}}^{|\mathcal{F}|}$$

where $\Gamma()$ is the Gamma function. If the data distribution in $\Omega(D)$ was completely uniform, then its density would be equal to: $|D|/vol(\Omega(D))$, where $vol(\Omega(D))$ denotes the volume of the smallest hyper-cube containing the whole allowed data space. For a particular test instance with (x_i, k_i), we desire to have k_i instances in its hypersphere so that the conversion from a k-NN instance to an r-N instance is accurate. Thus, k_i becomes the *mass*. Setting mass = density × volume, we get:

$$k_i = \frac{|D|}{vol(\Omega(D))} \cdot \frac{\pi^{|\mathcal{F}|/2}}{\Gamma\left(\frac{|\mathcal{F}|}{2}+1\right)} \cdot r_{\text{UNIF}}^{|\mathcal{F}|}$$

Algorithm 6 Interactive algorithm to find radius

Input: Private training dataset D, set of labels $\mathcal{L}$, list of k-NN test instances $X = ((x_1, k_1), \ldots, (x_{|X|}, k_{|X|}))$, privacy budget for conversion $\hat{\varepsilon}$, candidate count n
Output: List of r-N test instances $X' = ((x_1, r_1), \ldots, (x_{|X|}, r_{|X|}))$
1: Initialize X' as empty list
2: **for** $i = 1$ to $|X|$ **do**
3: Let $r_{\text{UNIF}} = \sqrt[|\mathcal{F}|]{\frac{\Gamma(\frac{|\mathcal{F}|}{2}+1)\cdot k_i \cdot vol(\Omega(D))}{|D|\cdot\pi^{|\mathcal{F}|/2}}}$
4: Let the set of candidates be $R = \{o_1, \ldots, o_n\}$, where $o_j = \frac{2 \cdot j \cdot r_{\text{UNIF}}}{n}$
5: Define $c^{\sum}_{x_i,o_j} = \sum_{I \in \mathcal{L}} c_{x_i,o_j}(I)$
6: Define quality function $u(D, o_j) = -|c^{\sum}_{x_i,o_j} - k_i|$
7: For $o_j \in R$, set $r_i = o_j$ with prob. $\propto e^{\frac{\hat{\varepsilon} u(D,o_j)}{2|X|}}$
8: Add (x_i, r_i) to X'
9: **end for**
10: **return** X'

Solving for r_{UNIF}, we obtain the r_{UNIF} on line 3 of Algorithm 6. However, this r_{UNIF} is not exactly accurate due to two reasons: (1) It assumes a perfectly uniform data distribution, which is almost never the case. (2) It requires apriori knowledge of the number of training instances, i.e., $|D|$. Some works in differential privacy allow this to

be public information (e.g., Zhang et al. 2014b) but for our purposes, a crude estimate suffices. A small privacy budget can be allocated to obtain the a noisy ε-DP estimate for $|D|$ using the Laplace mechanism.

To account for these, we do not directly set the radius of a test instance r_i to be equal to r_{UNIF}. Instead, on line 4 of Algorithm 6 we initialize a set of candidates $R = \{o_1, \ldots, o_n\}$, such that each o_j is a possible candidate for r_i. The size of R is determined by n, which is an input parameter. We specify $o_j = \frac{2 \cdot j \cdot r_{\text{UNIF}}}{n}$, e.g., if we had $r_{\text{UNIF}} = 0.5$ and $n = 4$, we would build the set of candidates $R = \{0.25, 0.5, 0.75, 1.0\}$. The upper bound of $2 \cdot r_{\text{UNIF}}$ is a heuristic decision so that even if the data in the proximity of the test instance x_i is sparse, we can have a reasonably large (but not too large) radius r_i for that test instance. Next, on line 5, Algorithm 6 defines $c^{\sum}_{x_i, o_j}$ as the total number of training instances within the hypersphere with center x_i and radius o_j. On line 6, the quality function $u(D, o_j)$ is defined as $u(D, o_j) = -|c^{\sum}_{x_i, o_j} - k_i|$, which assigns a penalty to each $o_j \in R$ proportional to the difference between $c^{\sum}_{x_i, o_j}$ and k_i (recall that having k_i training instances in the proximity region is our end goal). Then, line 7 uses the exponential mechanism to probabilistically choose a candidate and assign it as r_i. This completes the conversion for one test instance (x_i, k_i).

A privacy budget of $\hat{\varepsilon}$ is given to Algorithm 6 as an input. The data access on line 7 is through the exponential mechanism, and is repeated for every test instance (i.e., a total of $|X|$ times). Each access uses a privacy budget of $\hat{\varepsilon}/|X|$, so that the sequential composition of all accesses satisfies $\hat{\varepsilon}$-DP.

Non-interactive algorithm The approach taken by this algorithm is to create a data-dependent *global* index structure such that all conversions can be performed using that index structure. For this, we turn to ε-DP *spatial decompositions*.

Given a set of instances D on a domain $\Omega(D)$, spatial decompositions aim to accurately and privately approximate the data distribution, i.e., the distribution of the training instances in D over $\Omega(D)$. Existing solutions in the literature hierarchically decompose $\Omega(D)$ into cells according to the counts of each cell—if a cell contains very few instances, then it is not beneficial to decompose it further; but if it contains many instances then we should further zoom into that cell. This intuition stems from the trade-off between two sources of error: (i) noise error, i.e., having many cells means we have to divide the privacy budget many times and add noise to each cell separately, and (ii) approximation error, i.e., having few cells means we end up with a very coarse understanding of the data distribution and we have to approximate the data distribution *within* a cell.

This trade-off in choosing the right number of cells in a spatial decomposition has fueled many recent works in ε-DP literature. Cormode et al. (2012) introduced private kd-trees, quad-trees and R-trees. Qardaji et al. (2013) showed that simple uniform or adaptive grids yield higher accuracy than the previous methods, especially on low-dimensional data. Su et al. (2016) extended this analysis to high-dimensional data and offered *extended uniform grids*. To the best of our knowledge, this is the state of the art method in building spatial decompositions on high-dimensional data. Thus, we employ it in our work.

Algorithm 7 Non-interactive algorithm to find radius

Input: Private training dataset D, set of labels $\mathcal{L}$, list of k-NN test instances $X = ((x_1, k_1), \ldots, (x_{|X|}, k_{|X|}))$, privacy budget for conversion $\hat{\varepsilon}$, increment parameter γ
Output: List of r-N test instances $X' = ((x_1, r_1), \ldots, (x_{|X|}, r_{|X|}))$
1: Let $\mathcal{U}$ denote the uniform grid built over D with privacy budget $\hat{\varepsilon}$
2: Initialize X' as empty list
3: **for** $i = 1$ to $|X|$ **do**
4: $r_i \leftarrow \gamma$
5: **while** r_i does not exceed $|\Omega(D)|$ **do**
6: mass $\leftarrow 0$
7: **for** each cell $C \in \mathcal{U}$ **do**
8: Let V denote the volume of intersection between C and the hypersphere with center x_i and radius r_i
9: mass $\leftarrow$ mass + $V \times \frac{|C|}{vol(\Omega(C))}$
10: **end for**
11: **if** mass $\geq k_i$ **then**
12: Add (x_i, r_i) to X'
13: **break**
14: **else**
15: $r_i \leftarrow r_i + \gamma$
16: **end if**
17: **end while**
18: **end for**
19: **return** X'

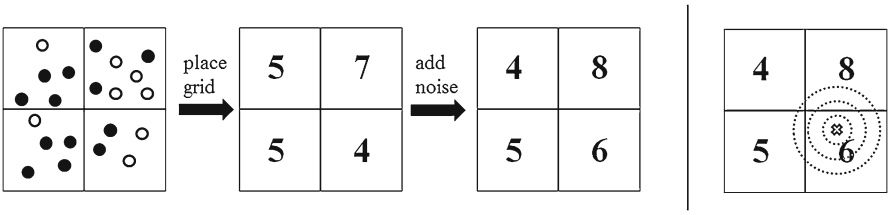

Fig. 6 Building a 2×2 noisy uniform grid (on the *left*) and conversion from k-NN to r-N using the grid (on the *right*)

The pseudocode for the non-interactive algorithm is presented in Algorithm 7. On the first line, we build $\mathcal{U}$, an $m \times m$ uniform grid on $\Omega(D)$. The choice of m should be dependent on D, $\Omega(D)$ and $\hat{\varepsilon}$, as demonstrated in Su et al. (2016). In general, m is positively correlated with $|D|$, $\hat{\varepsilon}$ and data dimensionality. For example, let the leftmost image in Fig. 6 depict the dataset and let $m = 2$. Then, the process of building the ε-DP uniform grid $\mathcal{U}$ is illustrated in Fig. 6. For each cell in the grid (i.e., $C \in \mathcal{U}$), we denote by $|C|$ the number of training instances in C, and by $vol(\Omega(C))$ the volume of C.

After the grid is built, we convert each k-NN instance to an r-N instance using the *for* loop between lines 3–18. For each instance $(x_i, k_i) \in X$, we place x_i on the grid as shown in the rightmost portion of Fig. 6. We initialize its radius $r_i = \gamma$, where γ is an input parameter to Algorithm 7. At each iteration of the *while* loop (lines 5–17), we increment r_i by γ and perform the following: We build the hypersphere with center x_i and radius equal to current r_i. Between lines 7–10, we find the expected

number of training instances in this hypersphere according to the grid. This is done by calculating the volume of intersection between the hypersphere and each cell, and multiplying this volume by the density of that cell. This inherently assumes that the data distribution within each cell is uniform. The total number of instances is stored in the variable called *mass*. Between lines 11–16, we check if the *mass* contains or exceeds k_i, the desired number of training instances in the hypersphere. If it does, then we stop the search and add (x_i, r_i) to X', and the conversion process for this test instance is complete. Otherwise, we continue by incrementing r_i by γ.

Comparison of the two algorithms The interactive and the non-interactive algorithms are essentially two alternatives that serve the same purpose: Converting a set of k-NN instances to a set of r-N instances, given a privacy budget $\hat{\varepsilon}$ for conversion. Although we experimentally compare them in Sect. 5, here we provide a verbal comparison.

First, the non-interactive algorithm relies on an index structure (in our case, a uniform grid) that approximates the distribution of the private data. The index structure is built independently of X (the set of test instances). As such, it does not take into account the distribution of the test instances. For instance, in Fig. 6 all test instances could be located in the lower right cell. In that case, it could be advantageous to zoom more in the lower right cell of the grid; but the grid does not consider this. Yet, being independent of X is not always disadvantageous—consider that after answering X, a new batch of test instances, $\hat{X}$, is issued. Then, the previously built grid can be re-used instead of building a new grid.

The main advantage of the interactive algorithm is that it directly takes into account the test instances in X within the conversion process. However, this algorithm is most suitable in cases where: (i) the non-interactive algorithm performs poorly, e.g., due to excessive data dimensionality, or very limited privacy budget that inhibits the creation of an accurate index structure, and (ii) the number of test instances, $|X|$, is small. Otherwise, if $|X|$ is high and $\hat{\varepsilon}$ is divided into many test instances, then each instance gets a tiny budget, thus the exponential mechanism on line 7 does not yield accurate results.

Finally, we note that both Algorithm 6 and Algorithm 7 are parametric. Algorithm 6 has a *candidate count* parameter denoted n and Algorithm 7 has an *increment* parameter denoted γ. Both n and γ control the trade-off between the accuracy and efficiency of the algorithms. In Algorithm 6, n should be high enough that at least some candidates in R are close to the actual r_i; but also small enough that R does not contain too many candidates (the latter would hurt efficiency). In our experiments, we saw that $n = 10$ often yields a good trade-off. In Algorithm 7, γ should be small enough such that r_i can be precisely found; but also high enough that the algorithm can find r_i efficiently (i.e., an excess number of iterations of the while loop on line 5 is not needed). In our experiments we often set γ to be approximately 0.01% of $|\Omega(D)|$.

5 Experimental evaluation

Implementation details We implemented our algorithms in Java. Experiments were conducted on a commodity laptop with Intel i7 2.40 GHz CPU and 16 GB main memory.

Evaluation metrics We use *classification accuracy* as our metric to quantify the utility of our algorithms. Classification accuracy measures the percentage of test instances that were correctly classified. Given a particular test setting (i.e., training dataset, test instances and ε) the classifier that yields the highest accuracy is the most desirable.

Datasets We obtained four numeric datasets from the UCI Machine Learning[2] and KEEL[3] repositories: *banana*, *phoneme*, *banknote* and *thyroid*. We only considered these numeric datasets since we can easily map records to Euclidean space and Euclidean distance (our choice of distance metric) is well-defined. Whereas for non-ordinal, categorical or text data, distance metrics are often subjective or application-dependent. We randomly divided each dataset into a private training dataset (containing 80% of the whole data) and test instances (containing the remaining 20% of the data). We used fivefold cross validation in our experiments. Next, we briefly explain our datasets.

The *banana* dataset contains 2 features (both continuous, real-valued), 2 labels and 5300 instances. It is an artificial dataset where instances belong to banana-shaped clusters. The labels represent the banana shapes in the dataset.

The *phoneme* dataset contains 5 features (all continuous), 2 labels and 5404 instances. It is a real-world dataset. The classification task is to distinguish between nasal sounds and oral sounds, given five phonemes as features.

The *banknote* dataset contains 4 features (all continuous), 2 labels and 1372 instances. Its classification task is to distinguish between genuine and forged banknote specimens. The features consist of information extracted from the images of the specimens.

The *thyroid* dataset contains 21 features (15 binary, 6 continuous), 3 labels and 7200 instances. Its classification task is to determine whether a patient referred to a clinic is hypothroid. The 3 labels signify normal (not hypothyroid), hyperfunction and subnormal functioning. The features include health-related readings from patients, e.g., age, sex, pregnancy status, past thyroid surgeries.

In addition to implementing our algorithms, we obtained the implementation of *PrivBayes* (Zhang et al. 2014b) from its authors.[4] PrivBayes is a state of the art differentially private data publishing algorithm. We compare with PrivBayes, since instead of running a k-NN classifier on private data, the data owner may publish his data first and perform classification on the published data. Our private k-NN classification algorithms should outperform this approach so that they remain useful wrt the state of the art. We choose PrivBayes rather than the other data publishing algorithms, because: (i) its authors experimentally show that they outperform prior work in ε-DP data publishing, and (ii) unlike most other work, PrivBayes does not require data to be discretized. The latter implies that PrivBayes can work with co-existing real and discrete-valued attributes. This is critical for meaningful k-NN classification with Euclidean distance. Since ε-DP is probabilistic, and so is PrivBayes; in each experiment we run PrivBayes 5 times and average its results.

[2] http://archive.ics.uci.edu/ml/datasets.html.

[3] http://sci2s.ugr.es/keel/datasets.php.

[4] https://sourceforge.net/projects/privbayes/.

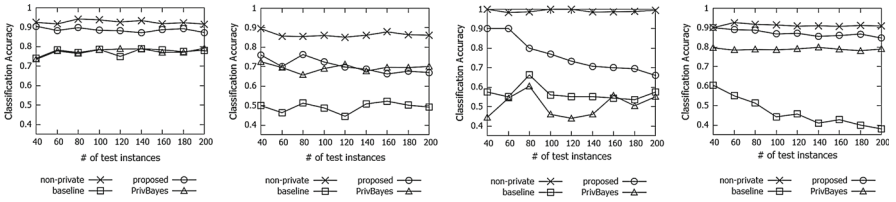

Fig. 7 r-N classification on banana, phoneme, banknote and thyroid datasets, with $\varepsilon = 0.5$

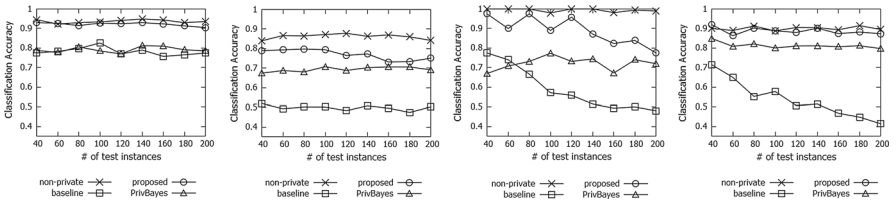

Fig. 8 r-N classification on banana, phoneme, banknote and thyroid datasets, with $\varepsilon = 1.0$

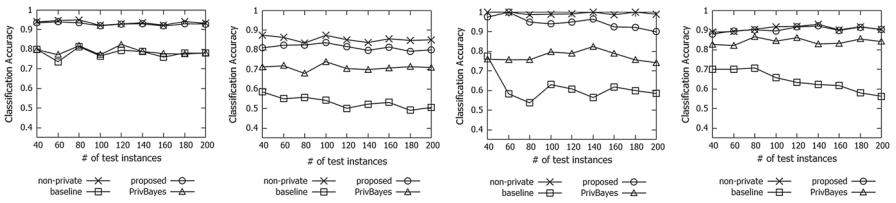

Fig. 9 r-N classification on banana, phoneme, banknote and thyroid datasets, with $\varepsilon = 2.0$

5.1 Evaluation of r-N classifiers

First, we evaluate our algorithms for differentially private r-N classification. For each of the datasets, we set the radius r so that on average, a test instance would have between 5 and 20 training instances in its proximity region.

We illustrate the results in Figs. 7, 8 and 9, for $\varepsilon = 0.5$, 1.0 and 2.0 respectively. In these figures, we denote by *non-private* the actual classification result without privacy, *baseline* the baseline ε-DP classifier presented in Sect. 3.1, *proposed* the proposed ε-DP classifier presented in Sect. 3.2, and *PrivBayes* the classification after ε-DP data publishing approach.

In Figs. 7, 8 and 9, we show how classification accuracy changes according to the number of test instances. Our motivation in Sect. 3.1 was that as we have more test instances, our budget per instance gets smaller, and thus we suffer from an over-division of the privacy budget. The results agree with this motivation, especially when $\varepsilon = 0.5$. When our budget per instance is small, the baseline solution has only 50% accuracy on the *phoneme* and *banknote* datasets, and approaches 33% accuracy on the *thyroid* dataset. Recalling that *phoneme* and *banknote* have 2 labels and *thyroid* has 3, we can conclude that the baseline solution's behavior approaches a random guess. Our proposed solution is also affected by an increase in the number of test instances, but not as much as the baseline solution. Even though the proposed solution performs

sensitivity analysis, more test instances means higher probability of intersection, thus higher sensitivity. As a result, some adverse effect is expected. These effects are visible when $\varepsilon = 0.5$ and 1.0 (e.g., on *banknote*), but almost negligible when $\varepsilon = 2.0$. On the other hand, since PrivBayes performs a one-time publishing of the private data, its classification accuracy is not affected by the number of instances.

Overall, the proposed solution clearly outperforms the baseline solution and PrivBayes. With $\varepsilon = 1.0$, on the average we obtain 5-10% accuracy loss (discrepancy with the non-private classifier) using the proposed solution. Using the baseline solution or PrivBayes, this amount is usually doubled or tripled. $\varepsilon = 1.0$ and 2.0 are reasonable values in ε-DP machine learning, considering that in many recent works, experiment results are given with $\varepsilon = 1.0$ to 4.0. (Recall that lower ε yields better privacy protection by the definition of ε-DP.) In that regard, we achieve good accuracy under strict privacy. Finally, we note the increase in classification accuracy as ε is increased, e.g., on the *banknote* dataset, accuracy is roughly 0.75 when $\varepsilon = 0.5$. It increases to 0.85 and 0.9-0.95 respectively when $\varepsilon = 1.0$ and $\varepsilon = 2.0$ using the proposed solution.

To validate the statistical significance of these findings, we also analyzed the confidence intervals, ranges and variances of the classifiers' accuracy. For this purpose, we repeated each experiment 20 times and obtained 95% confidence intervals. The non-private and proposed private classifiers have small standard deviation (roughly 0.03) and low margins of error, e.g., accuracy of the non-private classifier on *banana* is 0.94 ± 0.015 and the proposed classifier is 0.9 ± 0.016. PrivBayes has accuracy 0.77 ± 0.013. However, the baseline classifier has accuracy 0.78 ± 0.024, and has almost double the standard deviation. Hence, it can be deduced that the baseline classifier is more sensitive to randomization in the experiment setup. The margins of error differ slightly on other datasets, e.g., on *thyroid* they increase by roughly 0.01 for all classifiers. Given that the proposed method often has more than 0.1–0.15 accuracy gain compared to the baseline and PrivBayes, such small error margins imply strong statistical confidence that the proposed algorithm indeed outperforms its competitors.

5.2 Evaluation of k-NN to r-N conversion

An important step in our approach for k-NN classification is to privately convert a k-NN test instance to an r-N instance. For this purpose, we proposed the interactive algorithm (Algorithm 6) and the non-interactive algorithm (Algorithm 7). We compare the two under different conditions (i.e., different datasets and conversion budget $\hat{\varepsilon}$). For completeness, we also convert instances using the output of PrivBayes. We use two evaluation metrics: The first metric measures error in terms of the conversion result r, and the second metric measures error in terms of the discrepancy in k caused by an erroneous r.

Given a list X of k-NN instances, we first convert them to r-N instances non-privately: For each $(x, k) \in X$, we set its $r = \delta(x, x^k)$, where $\delta(x, x^k)$ is the distance between x and its kth nearest neighbor. These r values constitute our ground truth. Then, we apply our private conversion algorithms (i.e., Algorithm 6, 7 and PrivBayes) to obtain r', a noisy radius, for each $(x, k) \in X$. We compute the percentage of average

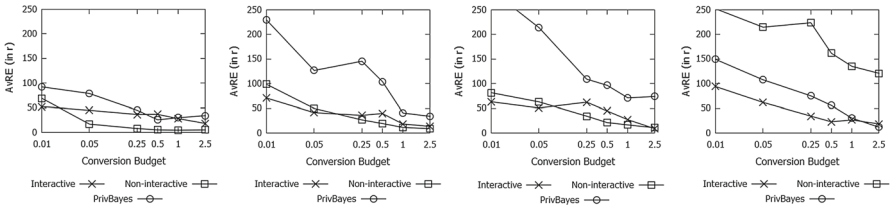

Fig. 10 Conversion from k-NN to r-N on banana, phoneme, banknote and thyroid datasets, with $k = 30$. Error measured using AvRE in r

relative error (AvRE) in r as follows:

$$\text{AvRE} = \frac{\sum_{(x,k)\in X} \frac{|r'-r|}{r} \cdot 100\%}{|X|}$$

An erroneous radius will also yield a discrepancy in k. Recall that $c^{\sum}_{x,r}$ denotes the number of training instances t in the dataset such that $\delta(x,t) \leq r$. Then, given that we would like to perform classification with k nearest neighbors, the non-private conversion yields $c^{\sum}_{x,r} = k$. However, since we obtain r' instead of r, we will perform classification with noisy k', where $c^{\sum}_{x,r'} = k'$. Our second metric below measures how much k' deviates from k using the Root Mean Square Error (RMSE). The reason we opt for RMSE instead of AvRE is because RMSE assigns a higher (super-linear) penalty to a higher discrepancy.

$$\text{RMSE} = \sqrt{\frac{\sum_{(x,k)\in X} \left(c^{\sum}_{x,r} - c^{\sum}_{x,r'}\right)^2}{|X|}}$$

The results with respect to the AvRE in r is given in Fig. 10, and the results with respect to RMSE in k is given in Fig. 11. The AvRE is high when $\hat{\varepsilon}$ is as low as 0.01, but as we move to more realistic $\hat{\varepsilon}$ (e.g., 0.5 or 1) at least one of our algorithms (interactive and non-interactive) yields below 30% error an all datasets. With $\hat{\varepsilon} = 1$ we achieve AvRE as low as 10–20%. The exception to this is the *thyroid* dataset which has 21 features. This causes high dimensionality, yet the number of instances is relatively low. As such, it suffers from the curse of dimensionality especially for the non-interactive algorithm: The grid ends up being too coarse, dividing data space into very few cells. A coarse grid does not suffice to understand the underlying data distribution well, thus the non-interactive algorithm performs particularly bad on this dataset. However, on the remaining 3 datasets, the non-interactive algorithm outperforms the interactive algorithm and PrivBayes. Both of our algorithms are often better than PrivBayes.

The results we have in Fig. 11 agree with the results in Fig. 10. In particular, when one algorithm performs better than the other wrt to r, it also performs better wrt to k. Thus, our discussion regarding which algorithm beats the other applies to this figure as well. On the other hand, the super-linear increase in RMSE becomes apparent with $\varepsilon \leq 0.05$. When the AvRE in $r \geq 70$–100%, the RMSE in k increases significantly; whereas for AvRE $\leq 50\%$, RMSE shows an almost linear trend.

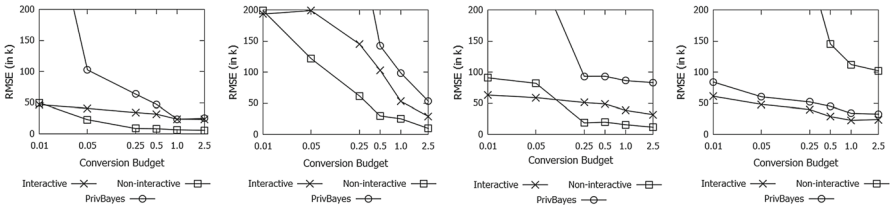

Fig. 11 Conversion from k-NN to r-N on banana, phoneme, banknote and thyroid datasets, with $k = 30$. Error measured using RMSE in k

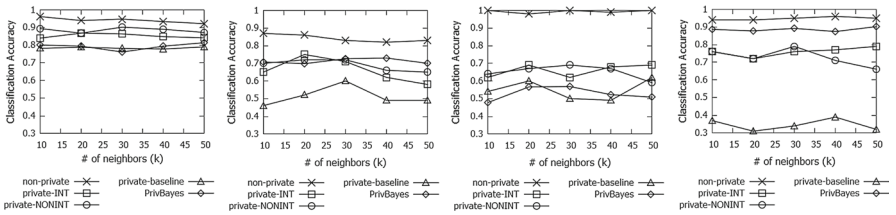

Fig. 12 k-NN classification on banana, phoneme, banknote and thyroid datasets, with $\varepsilon = 0.5$

The conclusions we draw from these experiments are twofold: (1) If data dimensionality is not high, for realistic $\hat{\varepsilon} = 0.5, 1.0, 2.0$, the non-interactive algorithm is the better choice. (2) For realistic $\hat{\varepsilon}$, our conversion is reasonably accurate, with relative errors of 20% or less in many cases.

5.3 Evaluation of k-NN classifiers

Next, we evaluate our algorithms for differentially private k-NN classification. In Sect. 5.1 we fixed r and varied the number of test instances. Orthogonal to that, in this section we fix the number of test instances to 100 and vary k, the number of neighbors.

We illustrate the results in Figs. 12, 13 and 14. In these figures, *non-private* denotes the results of k-NN classification without ε-DP, *private-INT* denotes the results of Algorithm 5 with Algorithm 6 (our proposed solution with interactive algorithm for conversion), *private-NONINT* denotes the results of Algorithm 5 with Algorithm 7 (our proposed solution with non-interactive algorithm for conversion), *private-baseline* denotes the results of baseline ε-DP k-NN classification (in Sect. 4.1), and *PrivBayes* denotes the results of publishing the data with PrivBayes and performing k-NN classification on the published data.

First, we observe that non-private k-NN classification is accurate on all four datasets (accuracy is higher than 0.85). Thus, k-NN is a good fit for these datasets. Then, we observe that when $\varepsilon = 0.5$, our algorithms perform better than PrivBayes on *banana* and *banknote*, head-to-head on *phoneme*, and worse on *thyroid*. However, when $\varepsilon = 1.0$ or 2.0, at least one of *private-INT* or *private-NONINT* significantly outperforms PrivBayes. Note that the *thyroid* dataset is naturally challenging due to its high dimensionality. The high errors in k-NN to r-N conversion (described in the previous section) play a role in this section: When conversion is inaccurate, classifi-

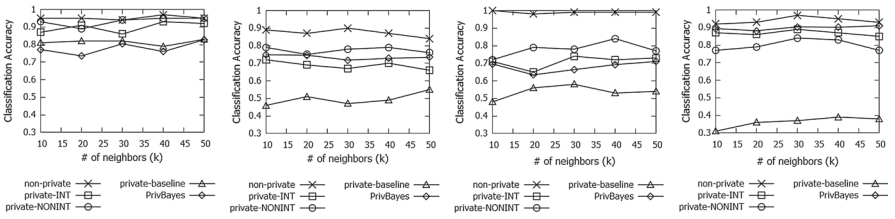

Fig. 13 k-NN classification on banana, phoneme, banknote and thyroid datasets, with $\varepsilon = 1.0$

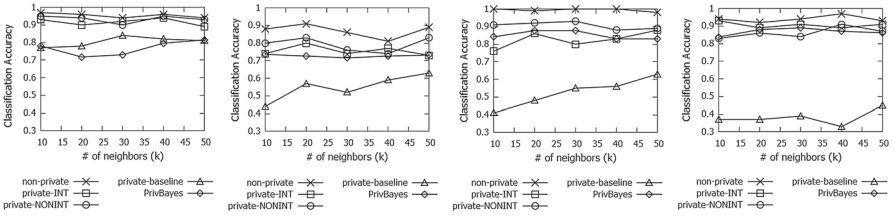

Fig. 14 k-NN classification on banana, phoneme, banknote and thyroid datasets, with $\varepsilon = 2.0$

cation also becomes inaccurate. Lowest conversion errors are obtained on the *banana* dataset, which also happens to have the largest discrepancy between our proposed algorithms and PrivBayes. The baseline approach performs poorly on all datasets, which motivates the need for advanced algorithms for ε-DP k-NN classification.

The margins of error (with 95% confidence intervals) are similar to those of the r-N classifiers. The non-private and non-interactive private algorithms have the lowest variances and margins of error (roughly ± 0.015), followed by the interactive and baseline algorithms (roughly ± 0.018). PrivBayes shows remarkably higher variance, and its margins of error are ± 0.04 on average.

5.4 Comparison with other classification techniques

Finally, we compare our ε-DP k-NN classification algorithms with other private classification techniques from the literature. We divide existing classification works into 3 broad categories: (i) linear classifiers, e.g., regression, SVM, perceptron, (ii) Bayesian classifiers, and (iii) decision trees and forests. We choose one popular work from each category. From (i), we choose the state of the art private logistic regression algorithm by Zhang et al. (2012). From (ii), we choose Naive Bayes Classifiers, which are widely used in recent works in ε-DP, e.g., Kotsogiannis et al. (2017), Zhang et al. (2016) and Xiao et al. (2011). From (iii), we choose the seminal work by Friedman et al. (2010) on ε-DP versions of ID3 and C4.5 algorithms for decision tree building. We use the C4.5 algorithm since it can handle features with continuous domains. For k-NN classification, we choose the better performing settings (in terms of k and conversion algorithm) based on the results reported in the previous sections.

We vary the privacy budget $\varepsilon \in [0.5, 2.0]$ and graph the results in Fig. 15. Our k-NN algorithm outperforms existing works completely on *banana*. On *phoneme*, it outperforms existing works when $\varepsilon \leq 1.25$, but is beaten by decision trees when

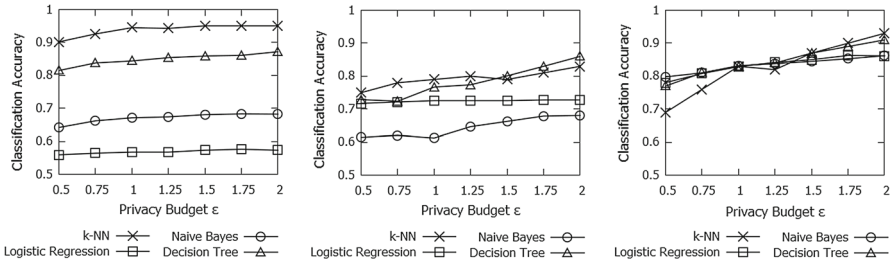

Fig. 15 Comparison of different differentially private classification techniques on banana, phoneme and banknote datasets

$\varepsilon \geq 1.5$. On *banknote*, it performs poorly when ε is low, but performs better than the other algorithms when $\varepsilon \geq 1.5$. We note that on *banknote*, most algorithms have similar classification accuracy.

It is generally accepted that certain classification techniques fit certain datasets very well, but may perform poorly on others. For example, observe that logistic regression performs poorly on *banana*. When analyzing the reason for this, we found that non-private logistic regression also performs poorly on *banana*, because the data points in this dataset consist of two-banana shaped, interleaved clusters. Therefore any linear classifier would yield low accuracy.

It is also interesting to quantify the accuracy differences between the private and non-private versions of different classification techniques. We found that these differences show variability depending on the dataset and ε. For example, the accuracy of decision trees drop by 2–10% on *banana*, but by 5–20% on *phoneme* and *banknote*. Therefore, even though non-private decision trees beat non-private k-NN, private k-NN performs better than private decision trees in many cases. In light of these results, choosing the right ε-DP classification technique apriori (depending on ε, characteristics of the dataset etc.) becomes an open question. We leave this to future work. We note, however, that ε-DP k-NN indeed outperforms its competitors in many settings, therefore deserving to be a viable option.

We again discuss the statistical significance of our results. Logistic regression, Naive Bayes and decision trees all build explicit models that are used for classification. In contrast, k-NN compares test instances directly with the training instances without building an explicit model. As a result, the other methods are more resilient to randomness caused by ε-DP noise and cross-validation. For example, the 95% confidence intervals for logistic regression and Naive Bayes indicate error margins of only ± 0.003, and the variance in their classification accuracy is much smaller. This number is increased fivefold or sixfold in the case of k-NN.

6 Related work

Differential privacy (DP) Differential privacy was introduced by Dwork (2006), and has gained significant attention since. We first discuss the most influential advances in DP. For queries with real-valued outputs, the Laplace mechanism was shown to

achieve DP (Dwork 2006). Even though this result was initially only for count queries, Dwork et al. extended the Laplace mechanism to functions like sums, linear algebraic functions and distance measures (Dwork et al. 2006). Later, for queries with integer-valued outputs, the geometric mechanism was proposed in Ghosh et al. (2012). Another improvement is due to McSherry and Talwar (2007) through the introduction of the exponential mechanism. The exponential mechanism can handle queries whose responses are members of arbitrary sets, which is especially useful for mechanism design. McSherry (2009), proved the composability of multiple DP mechanisms, i.e., the *sequential* and *parallel* composition properties. We formally define these advancements in Sect. 2 and make use of them throughout our work.

Research efforts in DP are being devoted to various tasks, including accurately answering range-count queries (Hay et al. 2016; Li et al. 2014), synthetic data and histogram publishing (Zhang et al. 2014a, b) and data mining and machine learning. Since the latter is most related to our work, we discuss it in detail below. For a more general survey, we refer the reader to Leoni (2012).

Data mining and machine learning with DP We divide this portion into: (i) supervised learning, (ii) unsupervised learning and other relevant ML techniques.

The work of Friedman et al. (2010) on building private decision trees using ID3 has sparked a plethora of work on differentially private random decision trees and forests. Several other approaches can be found in Bojarski et al. (2014) and Rana et al. (2015). Zhang et al. (2016) study Bayesian inference under DP, and Vaidya et al. (2013) build private Naive Bayes classifiers. Rubinstein et al. (2012) study private Support Vector Machines (SVMs). Zhang et al. (2012) propose the *functional mechanism* for linear and logistic regression. Chaudhuri and Monteleoni (2009) claim that logistic regression through sensitivity analysis causes excess noise, and instead they propose regression models based on objective perturbation. In comparison, our experimental results show that a tight bound yielding from sensitivity analysis can achieve accurate k-NN classifiers. Hamm et al. (2015) employ differential privacy to mine crowdsensed data.

In terms of unsupervised learning, most efforts have focused on clustering data with DP, and the k-means algorithm in particular. McSherry's PINQ (McSherry 2009) contains a private k-means implementation. Nissim et al. (2007) proposed k-means for a relaxation of ε-DP (namely, (ε, δ)-DP). Su et al. (2016) compare the accuracy of various private k-means implementations and propose the current state of the art approach. In addition to clustering, there have been efforts in private feature selection (Stoddard et al. 2014), outlier detection (Okada et al. 2015) and principal component analysis (Chaudhuri et al. 2013). More recently, deep learning with DP was discussed by Abadi et al. (2016) and Shokri and Shmatikov (2015). Ji et al. (2014) and Sarwate and Chaudhuri (2013) survey the studies on differentially private machine learning.

Privacy-preserving k-NN To the best of our knowledge, there is no previous work on differentially private k-NN classification or instance-based learning in general. Rather, we survey: (i) distributed private k-NN protocols, and (ii) k-NN on encrypted (outsourced) data.

In (i), data is horizontally partitioned among multiple owners. The honest-but-curious owners engage in a protocol to classify an incoming test instance, but wish not

to disclose any private information apart from what is needed to classify that instance. The general idea behind these protocols is to iteratively reveal neighbors closest to the test instance until a total of k neighbors are revealed. Prominent works in this area are by Kantarcioglu and Clifton (2004), Qi and Atallah (2008), Xiong et al. (2006), Xiong et al. (2007) and Zhang et al. (2009).

In (ii), the goal is to outsource k-NN computation and classification to the cloud or some untrusted third party. Since the data owner is trusted but the third party is untrusted, the data owner should be able to observe the results of the classification, but the third party should not. Works in this area employ cryptographic techniques (e.g., homomorphic encryption and garbled circuits). For instance, the data is encrypted before it is given to the third party, the third party performs k-NN search on encrpyted data and returns the results, and finally the data owner decrypts the results to obtain the classification result. Prominent works in this area are by Wong et al. (2009), Zhu et al. (2013), Elmehdwi et al. (2014), Yao et al. (2013) and Li et al. (2015).

7 Discussion and open problems

Comparison with data publishing Recall that in Sect. 5, we compared our private r-N and k-NN classification algorithms with PrivBayes, i.e., building classifiers on differentially private publication of the data. The motivation was that our algorithms should have better accuracy than classifiers built on published data; otherwise a one-time publication of the data would enable researchers to freely use the published data for any desired purpose, including (but not confined to) r-N and k-NN classification. Next, we give the advantages and disadvantages of using our proposed algorithms versus building classifiers on DP publication of the data.

As the experiments showed, our proposed algorithms had roughly 10–15% higher accuracy than classifiers built on DP published data. Clearly, our algorithms are specialized for r-N and k-NN classification, whereas data publishing is general purpose and task-agnostic. Therefore this increase in accuracy is justified. Data publishing has three additional limitations: (1) Our algorithms are typically fast, and can classify several hundreds of test instances within a few seconds. In contrast, data publishing can take several minutes or hours (Zhang et al. 2014b). (2) A problem that is specific to instance-based learning with numeric data and Euclidean distance is that most works in DP data publishing assume features with small, discrete domains, e.g., the case with binary features is very common. (PrivBayes was one of the few works that did not make this assumption.) It is unclear how these works would be employed to publish features with large, numeric, continuous domains. (3) The training data might be dynamic and new training instances may be added over time. A previously published dataset will not contain the new instances, whereas our private algorithms can take into account the new instances in evolving datasets, if employed in a scenario such as Fig. 1.

On the other hand, performing one-time DP data publishing can be preferable in several aspects: (1) The published data can be used for purposes other than instance-based learning, such as building other ML models, performing correlation analyses, clustering, etc. (2) Our r-N and k-NN algorithms require the queriers (i.e., the data

analysts) to spend their privacy budgets, which depletes their budget over time. In contrast, in data publishing the budget is spent once and a dataset is obtained in return, which can be used without any limitations or further spending of the budget. This is enabled by the post-processing property of DP. (3) Our proposed algorithms rely on the data owner being online and available to perform classification on a querier's demand. Since publishing is a one-and-done process, it does not require the data owner to be continuously available.

Given these advantages and disadvantages, we acknowledge that both data publishing and the proposed classification techniques have their place. The choice of using one over the other ultimately depends on the real-life setting, and preferences and priorities of the involved parties (e.g., the data owner and queriers). Furthermore, we believe that many of the issues raised above are not specific to DP r-N and k-NN classification, but instead they are shared by other DP machine learning algorithms as well.

Availability of test data in a batch When designing our r-N and k-NN classifiers, we assumed that the test instances X are available in batches, e.g., $X = ((x_1, k_1), \ldots, (x_n, k_n))$. This is a standard assumption in the ε-DP literature for many purposes such as query answering, data publishing and machine learning (Leoni 2012; Li et al. 2014). If queries are posed continuously or one-at-a-time, then the data owner has no knowledge of "what comes next", and optimization (sensitivity analysis or other forms of optimization) becomes much more challenging, if not impossible. We do emphasize that our classifiers can work in real-time and for continuous k-NN query evaluation, simply assuming $|X| = 1$. However, their behavior would converge to that of the baseline solutions, and we would not enjoy the accuracy benefits of our proposed solutions. Therefore we recommend answering in batch mode. This is not completely implausible, e.g., in collaborative environments, multiple queriers can combine their queries and ε budgets into a batch, and send the batch to the data owner. Alternatively, in environments that do not require immediate answering of queries, either the queriers or the data owner can accumulate several queries, treat them as a batch, and pose/answer them all at once.

Privacy of the test data One open problem for future research is the privacy of the test data. As shown in Fig. 1, the data owner is only responsible from protecting the training data. Unlabeled test instances that are shared with the data owner in our protocols are not a part of this data. That is why, the queriers may not completely trust the data owner, and they may wish to protect the privacy of their test instances from the data owner.

First, note that since r-N and k-NN are *instance-based* learning algorithms, they do not build explicit models. (In contrast, the likes of logistic regression and decision trees do.) Therefore, queriers are "forced" to share their test instances with the data owner so that classification can be performed. In this sharing, privacy can be accommodated in several ways, such as perturbation and secure multiparty computation (SMC). Several methods using SMC were surveyed in Sect. 6, under the descriptive name "privacy-preserving k-NN". Our algorithms can be used in combination with SMC to achieve the privacy of test instances. However, this combination would likely yield additional accuracy and/or efficiency loss, since SMC operations are computationally costly. Hence, we leave it for future work.

In addition, the collaborative environment setting above yields further interesting privacy properties. For example, if several queriers collaborate to build a batch of test instances, this may help disguise their queries by offering "crowd-blending privacy". That is, upon receiving the combined set of queries, the data owner may not trace which query originated from which querier. Such collaboration relies on SMC, and can be achieved with or without the existence of a trusted third party. However, note that this setting is not robust to malicious parties. For example, a malicious querier can pose many queries with large regions, which increases the maximum clique sizes in the region overlap graph, which in turn causes higher sensitivity and more noise in query answers.

Finally, we discuss if an honest-but-curious party (say, Alice) can make inferences regarding other queries and queriers in such a collaborative setting. We argue that in this case, probability of inference depends on Alice's existing background knowledge—if Alice has zero prior knowledge about the training data, then inferences are not likely. However, consider that Alice has background knowledge that she is querying a small, outlier region and she already knows the true label of her test instance. Assume that the noisy, ε-DP label that is returned to Alice does not match the true label she was expecting. Then, Alice infers that a large amount of noise must have been added to the answer, which must have been because many other queriers were querying the same region. In this case, Alice has made a clear inference regarding other queriers' queries. This situation is caused, in part, by the fact that differential privacy does not guarantee absolute confidentiality or absolute disclosure prevention—it is accepted that a party with more background knowledge will be able to make more inferences (Dwork and Naor 2008).

8 Conclusion

In this paper, we studied two instance-based classifiers (r-N and k-NN). We argued and experimentally showed that their baseline ε-DP implementations lead to undesirable loss in classification accuracy. Thus, we proposed more sophisticated algorithms to implement them. Our algorithm for private r-N classification was based on sensitivity analysis: We build a region overlap graph, from which we can find disconnected subgraphs that enjoy parallel execution, and further bound their sensitivity according to their maximum clique size. We showed theoretically that our analysis yields a tight upper bound on the amount of noise required to satisfy ε-DP. We then proposed private k-NN classifiers, based on the idea of converting them into r-N classifiers. We gave two algorithms for conversion in Sect. 4.3. We experimentally showed in Sect. 5 that the non-interactive algorithm is generally preferable unless data dimensionality is exceedingly high or ε is restrictively small. In addition, experiments illustrate that our proposed algorithms significantly outperform the baseline solutions, as well as classification on general purpose ε-DP data publication and other ε-DP machine learning techniques.

References

Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L (2016) Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. ACM, pp 308–318

Aggarwal CC (2014) Instance-based learning: a survey. In: Aggarwal CC (ed) Data classification: algorithms and applications. CRC Press, pp 157–186

Alcala J, Fernandez A, Luengo J, Derrac J, Garcia S, Sanchez L, Herrera F (2010) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Mult. Valued Log. Soft Comput. 17(2–3):255–287

Behley J, Steinhage V, Cremers AB (2015) Efficient radius neighbor search in three-dimensional point clouds. In: 2015 IEEE international conference on robotics and automation (ICRA). IEEE, pp 3625–3630

Bentley JL (1975) Survey of techniques for fixed radius near neighbor searching (No. SLAC-186; STAN-CS-75-513). Stanford Linear Accelerator Center, California

Bojarski M, Choromanska A, Choromanski K, LeCun Y (2014) Differentially-and non-differentially-private random decision trees. arXiv:1410.6973

Chaudhuri K, Sarwate AD, Sinha K (2013) A near-optimal algorithm for differentially-private principal components. J Mach Learn Res 14(1):2905–2943

Chaudhuri K, Monteleoni C (2009) Privacy-preserving logistic regression. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) Advances in neural information processing systems 21. Curran Associates, Inc., pp 289–296

Cormode G, Procopiuc C, Srivastava D, Shen E, Yu T (2012) Differentially private spatial decompositions. In: 2012 IEEE 28th international conference on data engineering. IEEE, pp 20–31

Doherty KAJ, Adams RG, Davey N (2007) Unsupervised learning with normalised data and non-Euclidean norms. Appl Soft Comput 7(1):203–210

Dwork C (2006) Differential privacy. In: 33rd international colloquium on automata, languages and programming, part II (ICALP 2006), pp 1–12

Dwork C (2008) Differential privacy: a survey of results. In: Agrawal M, Du D, Duan Z, Li A (eds) Theory and applications of models of computation. TAMC 2008. Lecture notes in Computer Science, vol 4978. Springer, Berlin, Heidelberg, pp 1–19

Dwork C, Naor M (2008) On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. J Priv Confidentiality 2(1):8

Dwork C, McSherry F, Nissim K, Smith A (2006) Calibrating noise to sensitivity in private data analysis. In: Theory of cryptography conference. Springer, Berlin, pp 265–284

Elmehdwi Y, Samanthula BK, Jiang W (2014) Secure k-nearest neighbor query over encrypted data in outsourced environments. In: 2014 IEEE 30th international conference on data engineering. IEEE, pp 664–675

Friedman A, Schuster A (2010) Data mining with differential privacy. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 493–502

Ghosh A, Roughgarden T, Sundararajan M (2012) Universally utility-maximizing privacy mechanisms. SIAM J Comput 41(6):1673–1693

Hamm J, Champion AC, Chen G, Belkin M, Xuan D (2015, June). Crowd-ML: a privacy-preserving learning framework for a crowd of smart devices. In: 2015 IEEE 35th international conference on distributed computing systems (ICDCS). IEEE, pp 11–20

Hay M, Machanavajjhala A, Miklau G, Chen Y, Zhang D, Principled evaluation of differentially private algorithms using DPBench. In: Proceedings of the 2016 ACM SIGMOD international conference on management of data

Horton P, Nakai K (1997) Better prediction of protein cellular localization sites with the k nearest neighbors classifier. In: Proceedings of the 5th international conference on intelligent systems molecular biology, vol 5, pp 147–152

Ji Z, Lipton ZC, Elkan C (2014) Differential privacy and machine learning: a survey and review. arXiv:1412.7584

Kantarcioglu M, Clifton C (2004) Privately computing a distributed k-nn classifier. In: European conference on principles of data mining and knowledge discovery. Springer, Berlin, pp 279–290

Karp RM (2010) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of computer computations: proceedings of a symposium on the complexity of computer computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. Springer, Berlin, Heidelberg, pp 85–103

Kotsogiannis I, Machanavajjhala A, Hay M, Miklau G (2017) Pythia: data dependent differentially private algorithm selection. In: Proceedings of the 2017 ACM international conference on management of data. ACM, pp 1323–1337

Leoni D (2012, May) Non-interactive differential privacy: a survey. In: Proceedings of the first international workshop on open data. ACM, pp 40–52
Li C, Hay M, Miklau G, Wang Y (2014) A data-and workload-aware algorithm for range queries under differential privacy. Proc VLDB Endow 7(5):341–352
Li F, Shin R, Paxson V (2015) Exploring privacy preservation in outsourced k-nearest neighbors with multiple data owners. In: Proceedings of the 2015 ACM workshop on cloud computing security workshop, ACM, pp 53–64
Machanavajjhala A, Korolova A, Sarma AD (2011) Personalized social recommendations: accurate or private? Proc VLDB Endow 4(7):440–450
McSherry FD (2009) Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: Proceedings of the 2009 ACM SIGMOD international conference on management of data. ACM, pp 19–30
McSherry F, Talwar K (2007) Mechanism design via differential privacy. In: 48th annual ieee symposium on foundations of computer science, 2007. (FOCS'07). IEEE, pp 94–103
Nissim K, Raskhodnikova S, Smith A (2007) Smooth sensitivity and sampling in private data analysis. In: Proceedings of the 39th annual ACM symposium on theory of computing, ACM, pp 75–84
Okada R, Fukuchi K, Sakuma J (2015) Differentially private analysis of outliers. In: Joint European conference on machine learning and knowledge discovery in databases. Springer International Publishing, pp 458–473
Parry RM, Jones W, Stokes TH, Phan JH, Moffitt RA, Fang H, Wang MD (2010) k-Nearest neighbor models for microarray gene expression analysis and clinical outcome prediction. Pharmacogenomics J 10(4):292–309
Qardaji W, Yang W, Li N (2013) Differentially private grids for geospatial data. In: 2013 IEEE 29th international conference on data engineering (ICDE). IEEE, pp 757–768
Qi Y, Atallah MJ (2008, June) Efficient privacy-preserving k-nearest neighbor search. In: 2008 IEEE 28th international conference on distributed computing systems (ICDCS). IEEE, pp 311–319
Rana S, Gupta SK, Venkatesh S (2015) Differentially private random forest with high utility. In: 2015 IEEE international conference on data mining (ICDM). IEEE, pp 955–960
Rubinstein BI, Bartlett PL, Huang L, Taft N (2012) Learning in a large function space: privacy-preserving mechanisms for SVM learning. J Priv Confidentiality 4(1):65–100
Sarwate AD, Chaudhuri K (2013) Signal processing and machine learning with differential privacy: algorithms and challenges for continuous data. IEEE Signal Process Mag 30(5):86–94
Scikit-learn: machine learning in python. http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.RadiusNeighborsClassifier.html. Retrieved Jan 20 2017
Shokri R, Shmatikov V (2015) Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. ACM, pp 1310–1321
Stoddard B, Chen Y, Machanavajjhala A (2014) Differentially private algorithms for empirical machine learning. arXiv:1411.5428
Su D, Cao J, Li N, Bertino E, Jin H (2016) Differentially private k-means clustering. In: Proceedings of the sixth ACM conference on data and application security and privacy. ACM, pp 26–37
To H, Ghinita G, Shahabi C (2014) A framework for protecting worker location privacy in spatial crowdsourcing. Proc VLDB Endow 7(10):919–930
Vaidya J, Shafiq B, Basu A, Hong Y (2013) Differentially private naive Bayes classification. In: Proceedings of the 2013 IEEE/WIC/ACM international joint conferences on web intelligence (WI) and intelligent agent technologies (IAT). IEEE, pp 571–576
Wong WK, Cheung DWL, Kao B, Mamoulis N (2009) Secure kNN computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD international conference on management of data. ACM, pp 139–152
Wu Q, Hao JK (2015) A review on algorithms for maximum clique problems. Eur J Oper Res 242(3):693–709
Wu X, Kumar V, Quinlan JR, Ghosh J, Yang Q, Motoda H, Zhou ZH (2008) Top 10 algorithms in data mining. Knowl Inf Syst 14(1):1–37
Xiao X, Bender G, Hay M, Gehrke J (2011) iReduct: differential privacy with reduced relative errors. In: Proceedings of the 2011 ACM SIGMOD international conference on management of data. ACM, pp 229–240
Xiong L, Chitti S, Liu L (2007) Preserving data privacy in outsourcing data aggregation services. ACM Trans Internet Technol (TOIT) 7(3):17

Xiong L, Chitti S, Liu L (2006) K nearest neighbor classification across multiple private databases. In: Proceedings of the 15th ACM international conference on information and knowledge management. ACM, pp 840–841

Yao B, Li F, Xiao X (2013) Secure nearest neighbor revisited. In: 2013 IEEE 29th international conference on data engineering (ICDE). IEEE, pp 733–744

Zhang J, Zhang Z, Xiao X, Yang Y, Winslett M (2012) Functional mechanism: regression analysis under differential privacy. Proc VLDB Endow 5(11):1364–1375

Zhang X, Chen R, Xu J, Meng X, Xie Y (2014) Towards accurate histogram publication under differential privacy. In: Proceedings of the 2014 SIAM international conference on data mining, pp 587–595

Zhang J, Cormode G, Procopiuc CM, Srivastava D, Xiao X (2014) Privbayes: private data release via bayesian networks. In: Proceedings of the 2014 ACM SIGMOD international conference on management of data, ACM, pp 1423–1434

Zhang Z, Rubinstein BI, Dimitrakakis C (2016) On the differential privacy of Bayesian inference. In: The thirtieth AAAI conference on artificial intelligence (AAAI-16)

Zhang F, Zhao G, Xing T (2009) Privacy-preserving distributed k-nearest neighbor mining on horizontally partitioned multi-party data. In: International conference on advanced data mining and applications, Springer, Berlin, pp 755–762

Zhu Y, Xu R, Takagi T (2013) Secure k-NN computation on encrypted cloud data without sharing key with query users. In: Proceedings of the 2013 international workshop on security in cloud computing. ACM, pp 55–60