LETEO: Scalable anonymization of big data and its application to learning analytics

ANII Fondo sectorial de investigación con de datos - 2018

Technical report

Scientific coordinators:

Dr. Eduardo Giménez (ICT4V)

Dra. Lorena Etcheverry (INCO,FING,UdelaR)

Researchers and technical staff:

Dr. Federico Olmedo (IMFD, Chile)

Dr. Carlos Buil Aranda (IMFD, Chile)

Dr. Matías Toro (IMFD, Chile)

Bch. Marcos Pastorini (INCO, FING, UDELAR)

October 2021

Summary

Created in 2007, Plan Ceibal is an inclusion and equal opportunities plan with the aim of supporting Uruguayan educational policies with technology. Throughout these years, and within the framework of its tasks, Ceibal has an important amount of data related to the use of technology in education, necessary to manage the plan and fulfill the assigned legal tasks. However, the data does not they can be studied without accounting for the problem of de-identifying the users of the Plan.

To exploit this data, Ceibal has deployed an instance of the Hortonworks Data Platform (HDP), a open source platform for the storage and parallel processing of massive data (big data). HDP offers a wide range of functional components ranging from large file storage (HDFS) to distributed programming of machine learning algorithms (Apache Spark / MLlib). However, as of today there are no solutions for the de-identification of personal code data open and integrated into the Hortonworks ecosystem. On the one hand, the de-identification tools existing data have not been designed so that they can easily scale to large volumes of data, and they also do not offer easy integration mechanisms with HDFS. This forces you to export the data outside of the platform that stores them to be able to anonymize them, with the consequent risk of exposure of confidential information. On the other hand, the few integrated solutions in the Hortonworks ecosystem are owners and the cost of their licenses is very significant.

The objective of this project is to promote the use of the enormous amount of educational and technological data that Ceibal possesses, lifting one of the greatest obstacles that exist for that, namely, the preservation of privacy and the protection of the personal data of the beneficiaries of the Plan. To this end, this project seeks to generate anonymization tools that extend the HDP platform. On In particular, it seeks to develop open source modules to integrate into said platform, which implement a set of programmed anonymization techniques and algorithms in a distributed manner using Apache Spark and that can be applied to data sets stored in HDFS files.

Keywords: anonymization, big data, learning analytics

Contents

1 Introduction

Plan Ceibal was created in Uruguay in 2007 and is a plan for inclusion and equal opportunities to support Uruguayan educational policies with technology. Throughout these years, Ceibal has collected a significant amount of data related to the use of technology in education, necessary for managing the plan and the fulfillment of the legal tasks assigned to it. The value of this data is evident when it comes to developing learning analytics, i.e., quantitative techniques based on the exploitation of these data to analyze and account for educational problems. The determination of user profiles of communication networks, the design of personalized learning trajectories, and the development of early dropout indicators and alerts are some examples of this type of analysis.

The application of machine learning (ML) algorithms, pattern recognition, anomaly detection, or natural language processing using neural networks open new perspectives to answer these questions using quantitative methods based on accumulated data. Indeed, these techniques require the availability of consistent volumes of data for the training of prediction and correlation algorithms, and Ceibal is in a position to perform these analyses. In addition, Ceibal is frequently asked by other organizations to provide them with datasets to carry out studies. However, the use of learning analytics raises legal, ethical, and technological issues. Therefore, it is essential to ensure that the conduct of these studies and the publication of the results provide the necessary guarantees to protect the privacy of the individuals involved and their data.

In this context, the need arises to apply automated solutions that make it possible to de-identify (anonymize) personal data before using them in studies, either within Ceibal or other organizations. These methods should transform data consistently and make it extremely difficult to re-identify individuals. This approach is known as privacy-preserving data publishing (PPDP) and has a relatively low impact on the work of data scientists. On the other hand, the notion of differential privacy, which views privacy as a property of systems rather than data, has gained traction in recent years. Following this second approach, it is possible to embed privacy in database systems or machine learning algorithms.

Beyond these two approaches, Ceibal poses technological constraints regarding the solutions to be used. Since 2018, Ceibal has an open-source Big Data Platform (Hortonworks Data Platform, HDP) that covers various aspects in the exploitation of such data: storage for large files (Hadoop/HDFS), computational parallelization and horizontal scaling (YARN/MapReduce), data access control (Apache Ranger), metadata tagging and data lineage (Apache Atlas), distributed programming, ML and AI libraries (Spark, MLlib, Apache Zeppelin).

This project explores the application of PPDP and differential privacy, assuring that data manipulation is performed within the HDP platform. Also, ensuring that the developed solutions scale appropriately for the massive volume of data that Ceibal has accumulated that will undoubtedly continue to grow at an even faster rate in the coming years.

This document is organized as follows: Section 2 presents the basic concepts on anonymization and differential privacy used in our work, including also possible use cases gathered during the project. Next, Sections 3 and 4 present the approaches presented before, includ-

ing experimental evaluation of them. Finally, in Section **??** we conclude the accomplished work and outline future perspectives.

2 Preliminary Concepts

Data privacy or information privacy is a part of data protection that deals with the proper treatment of data with a focus on compliance with data protection regulations. Data anonymization is one of the techniques that organizations can use to comply with data privacy regulations that require the security of personally identifiable information (PII), such as health reports, contact information, and financial details. This approach is particularly relevant when organizations decide to publish or release a dataset, which may expose PII.

2.1 Data anonymization: privacy as a data property

Data anonymization, also known as de-identification, is a data transformation process that seeks to reduce the risk of disclosing sensitive micro-data. This technique assumes that, instead of the original dataset *D* containing the personal data in detail, another dataset D^T is published, which is the result of applying some transformation f to each row of D. Typically, two different disclosure risks are considered. On the one hand, **identity disclosure** refers to an attacker's ability to deduce to which individual a published record belongs. This case is also known as the re-identification of a record. On the other hand, **attribute disclosure** risk refers to an attacker being able to determine, with a high level of security, the value of a confidential attribute of an individual. For example, an attacker would know with certainty that a student possesses a certain familial status.

Privacy models seek to reduce the risks described above and define a series of characteristics that data sets must satisfy to mitigate them. *k-anonymity*[8] is one of the simplest privacy models, and it's main goal is to reduce the re-identification risk. The idea underlying this model is to bound the probability for an attacker who knows a published record $f(r)$ can link it to its pre-image r in the original database that gave birth to it. More precisely, we say that the transformed dataset D^T is k-anonymous if each combination of values for a subset of attributes called *quasi-identifiers* appears repeated at least k times in D^T. If this is the case, the attacker is always confronted to at least k possible candidates in the original dataset that could be the pre-image of the target. Hence, the probability for the attacker to re-identify the person associated to the record is less than $1/k$.

3 Top-down k-anonymization using Spark

3.1 Preliminaries

Top-down specialization (TDS) is an algorithm for producing a k-anonymous dataset. It proceeds by generalization and replacement of the values in the original dataset from valus of another, enlarged dataset.

Let D be a relational dataset, made of tuples $\{a_1 := v_1, \dots a_m := v_m\}$ which provide values for m attributes $a_1, \dots a_m$. Each attribute a_i has an associated domain $\mathcal{D}(a_i)$ for its values. The dataset contain N tuples, also called rows, all with n columns and the same attributes. The attributes of D are classified into the following categories:

- **Identifiers** Attributes that directly identify an individual.
- **Quasi-identifiers** Attributes that, combined with other quasi-identifiers, from tuples that can be associated with a small number of individuals. We assume that the attacker may know the value of these attributes for the victim, and will use them to reidentify the tuple of the victim in the dataset.
- **Sensible attributes** Attributes that the attacker do not know and wants to disclose for the victim.
- **Other** Other attributes that do not fall in any of the precedent categories. They are either uninteresting for the attacker or we assume that the attacker does not know their value for the victim.

Let $q_1, \dots q_n$ be the quasi-identifiers of D. We note $D[q_1, \dots q_n]$ the dataset obtained by dropping from D all the attributes that are not in the subset $\{q_1, \dots q_m\}$, and $D[q_1 := v_1, \dots q_n := v_n]$ the dataset obtained by removing from $D[q_1, \dots q_n]$ the rows that do not take the values $q_1 := v_1, \dots q_n := v_n$.

As part of this anonymization technique, identifiers are either removed from the dataset or transformed using a hashing function. Quasi-identifiers are transformed into other, more generic values, in order to achieve k-anonymity. In order to introduces this concept, we first define what is the level of anonymity provided by a daatset:

Definition 3.1 (Level of anonymity of a dataset.) *We say that the level of anonymity of D is l if l is the minimum value of $|D[q_1 := v_1, \dots, q_m := v_m]|$ that can be obtained for any assignment of values $v_1, \dots v_n$ of its quasi-identifiers $q_1, \dots q_n$. We note the level of anonymity of D as $\mathcal{A}[D]$*

Based on this definition, we formally introduce the notion of k-anonymity as follows:

Definition 3.2 (k-anonymity) *We say that D is k-anonymous if the level of anonymity provided by D is greater or equal than k.*

We now turn to the description of an algorithm for transforming a dataset into a k-anonymous one.

3.2 Approach description

The goal of TDS is to generate another k-anonymous dataset D^T with the attributes $a_1^T \dots a_m^T$ such that for each quasi-identifier q_i of D, the elements of the domain $\mathcal{D}(q_i^T)$ denote sets

of elements of $\mathcal{D}(q_i)$, while any other attribute a_i of D remains unchanged and with the same domain. For example, a quasi-identifier of D representing the age of an individual as a positive integer less than 110 may give raise to a quasi-identifier which takes as values ranks of ages between 0 and 110, such as for instance, the rank $[25, 50]$.

In order to achieve this, we assume that each quasi-identifier q_i is equipped with a function $p_i : \mathcal{D}(q_i^T) \rightarrow List(\mathcal{D}(q_i^T))$ which provides a partition for any set of $\mathcal{D}(q_i)$, that is, a collection of subsets that are mutually disjoint and where the union is the given input set. Furthermore, we consider that the tree obtained by recursively applying p_i to each element in the partition $p_i(x)$ of x is well-founded. The TDS algorithm computes a partition for each quasi-identifier q_i, starting from the whole domain $\mathcal{D}(q_i)$ as initial approximation (which has to be k-anonymous for a solution to exist), and iterating the functions p_i until further specializing an element of the partition would entail losing k-anonymity. The transformed dataset D^T is then obtained mapping the quasi-identifiers of each row to the set in the corresponding partition. Notice that, as it is a partition, it classifies all elements of the domain $\mathcal{D}(q_i)$.

3.2.1 Internal state

The internal state of the TDS algoritm is therefore determined by two components: the current partition and a working dataset. The algorithm works iteratively, updating the current partition and using the working dataset to store intermediate results. Let us further detail each of these two components.

Current partition. This part of the internal state contains current partition that has been reached so far. This can be though of as a list of lists of sets, each list representing the current partition of one of the quasi-identifiers. For simplicity, this logical list of lists is actually represented as a flat list of pairs (q_i, X), where the first component of the pair is the quasi-identifier q_i and the second component is a class X of the partition of q_i.

Working Dataset. This is an auxiliary dataset D^* containing temporary calculations introduced for the sake of the sake of the transformation. In this dataset we kept both the original values of the quasi-identifiers $q_i \dots q_m$, the list of transformed quasi-identifiers $q_1^T \dots q_m^T$ with the corresponding set in the partition, and an additional list of so-called *candidate columns* q_i^X for each set (q_i, X) in the current partition. These candidate columns are computed by replacing any occurrence of the set X in the attribute q_i^T of the row D_j^* by the set $Y \in p_i(X)$ such that $D_j^*(q_i) \in Y$. In other words: q_i^X is the transformed column that we would obtain if (q_i, X) would be specialized.

The name of the attributes of D^* may therefore have no superinex at all (the original quasi-identifiers), the special superindex "T" (transformed column), or the a superindex that is the name of one of the sets in the current partition. These superindex are used create additional, auxiliary columns in the working dataset D^*. Moreover, we use the notation $q_h^{i \rightarrow X|T}$ to denote either the candidate attribute q_i^X of D^* if $h = i$ or q_h^T otherwise. Consistently, we use the notation to $D^*[i \rightarrow X|T]$ to denote the dataset $D^*[q_1^T, \dots q_{i-1}^T, q_i^X, q_{i+1}^T, \dots q_n^T]$ and $D^*[T]$ to denote $D^*[q_1^T, \dots q_n^T]$.

3.2.2 The algorithm

The TDS algorithm takes as inputs a dataset D, the set of its quasi-identifiers $\{q_1 \dots q_n\}$ and their associated partitioning functions $\{p_1 \dots p_n\}$. It can be sketched as follows:

1. **Initialization step.**
   - Initialize the current partition with the whole domain of each attribute $(q_i, \mathcal{D}(q_i))$ for $i = 1 \dots n$.
   - Create the working dataset D^* with the columns $q_i^T := \mathcal{D}(q_i)$ and the candidate columns $q_i^{\mathcal{D}(q_i)}$ resulting from replacing $\mathcal{D}(q_i)$ by the set $X \in p_i(\mathcal{D}(q_i))$ such that $D(q_i) \in X$.
   - Check that $D^*[q_1^T, \dots q_n^T]$ is k-anonymous. If it is not, then return an error: it is not possible to transform the dataset in order to obtain a k-anonymous one.
2. **Selection step.** Pick an element (q_i, X) in the current partition such that $p_i(X) = \{X_1, \dots X_o\} \neq \{\}$ and $D^*[i \to X|T]$ is k-anonymous. If no such element exists in the current partition, then stop and return the current partition. Otherwise:
3. **Specialization step.** Replace in the current partition the element (q_i, X) by the list $(q_i, X1), \dots (q_i, X_o)$. Add to the internal dataset D^* the (lazy) candidate columns $q_i^{X_1}, \dots q_i^{X_o}$ as explained.
4. **Iteration step.** Goto 2

Notice that the more the domains of the quasi-identifiers are specialized, the closer the transformed dataset will be to the original one, but the less anonymous it will be. The TDS algorithm searches for a compromise between the information and anonymity level.

This compromise is closely related with the selection step. It is clear that it may exist several candidate subsets (q_i, X) in the current partition that can be selected for specialization. Specializing a given subset of the partition restricts the choices of further possible candidates in a way that depends on the dataset. For example, if in the dataset the elements matching one of the children of X are such that they are all combined with less than k elements of attribute q_j, then choosing to partition (q_i, X) will prevent q_j to be partitioned in further iterations. This means that different choices of the subset to be specialized result in different solutions. Furthermore, choosing one candidate subset may also impact the speed with which the TDS algorithm converges to a solution. It is therefore important to establish a criterion for selecting the candidate for further specialization.

This problem is solved using a greedy approach. At each step, the candidate that offers the best compromise between information and anonymity is selected. This compromise is expressed using a gain function

$$\mathcal{G}(s) = \frac{\mathcal{G}_{\mathcal{I}}[s]}{1 + \mathcal{G}_{\mathcal{A}}[s]} \tag{3.1}$$

where $\mathcal{G}_{\mathcal{I}}[s]$ represents the information gain and $\mathcal{G}_{\mathcal{A}}[s]$ the anonymity gain when specializing the subset s of the current partition.

Let $s = (q_i, X)$. The definition of $\mathcal{G}_{\mathcal{A}}[s]$ is straightforward:

$$\mathcal{G}_{\mathcal{A}}[(q_i, X)] = \mathcal{A}[D^*[T]] - \mathcal{A}[D^*[i \rightarrow X|T]] \tag{3.2}$$

In words, it is defined as the difference between the current level of anonymity achieved so far, and the one that is achieved by specializing s.

On the contrary, several possible measures can be taken for the function $\mathcal{G}_{\mathcal{I}}[s]$. In this work we considered two of them: Shannon's information entropy and squared distance.

Shannon's information entropy is a measure of the *diversity* of values that can be obtained by specializing a set of the partition. Alternatively, it can also be thought of as a measure of the *uncertainty* of finding a particular value. The lower the entropy of s, the more biased are the specialization of s to one of its subsets, and therefore the more likely is to find their values in the dataset. In this sense, it provides fewer information that refining another subset w which can be partitioned into subsets such that their values are equally likely to appear in the dataset. This metric is formally defined as follows:

$$\mathcal{G}_{\mathcal{I}}[(q_i, X)] = \mathcal{E}(D^*[T]) - \mathcal{E}(D^*[i \rightarrow X|T]) \tag{3.3}$$

where $\mathcal{S}$ is the definition of Shannon entropy:

$$\mathcal{E}(D) = -\sum_{j=1}^{N} \mathsf{Pr}\left[D_j\right] * log_2(\mathsf{Pr}\left[D_j\right]) \tag{3.4}$$

where N the number of rows in D and $Pr(D - j)$ the ratio of occurrences of the row D_j in the whole dataset D.

Squared distance is based on the idea of focusing on the *accuracy* provided by the transformed dataste D^T. Accuracy is measured as the (square of) the *distance* from the value of the attribute in the original dataset to the representative of the corresponding partition to which it belongs in the transformed dataset. For using this metric, we assume that each quasi-identifier is equipped with a normalized distance $d_{q_i}(x, y) \in [0, 1]$ from a value $x \in \mathcal{D}(q_i)$ to its abstraction $y \in \mathcal{D}(q_i^T)$. In this case, we define the information gain as follows:

$$\mathcal{G}_{\mathcal{I}}[(q_i, X)] = \sum_{j=1}^{N} \mathsf{Pr}\left[D_j\right] * \sum_{h=1}^{n} \frac{dist^2(D_j(q_h), D_j(q_h^{i \rightarrow C|T})}{n} \tag{3.5}$$

where n is the number of quasi-identifiers, N the number of rows in D, $Pr(D - j)$ the ratio of occurrences of the row D_j in the whole dataset D.

The distance for each quasi-identifier can be easily defined in for numerical attributes. For categorical ones, we use the notions of distance on a finite domain with a lattice structure that are defined in [**?**].

3.2.3 Setup

The TDS algorithm was implemented in Spark/Cloudera,a platform for processing big data based on the Hadoop distributed file system. In Hadoop, the dataset is distributed among the working nodes of a cluster of servers. Spark provides the primitives for performing operations on the dataset following the map/reduce paradigm. The task on the dataset are spread among the servers of the cluster, each one performing the part of it that involves the piece of data it has. Spark also provides an SQL-like language in which one can describe operations such as group-by, counting, selecting and adding up columns, creating columns in a lazy way, etc.

The TDS algorithm was programmed in Scala, the native language of Spark. In the way it is described in this paper, the algorithm has a natural and elegant parallel implementation in Hadoop. The iterative part of TDS is executed in the head node, while the operation for computing the gain function and specializing the internal dataset are launched in parallel among the nodes of the cluster. All the parallel part of the algorithm is encapsulated in the primitives of the Spark library.

3.3 Experimental Evaluation

This technique was evaluated on the dataset that Ceibal used for a clustering experiment. The aim of the experiment was to determine user profiles among the CREA2 and PAM learning platforms. The input dataset included several quasi-identifiers (sex, educational system, course level, state, region), a sensitive attribute (family income quintil) and other type of numeric data regarding the number of times the studen connected to the platform. The dataset was transformed using the TDS algorithm and a taxonomy of values for each quasi-identifier, in order to produce a 10-anonymity dataset. As a result, it was found that the original dataset contained several unique tuples and combinations that occur less than 10 times. After curating the dataset and eliminating outilers, a transformed dataset was produced in which all the attributes can be specialized to the original values, except from the course level.

The performance (execution time) of the TDS algortihm in the Cloudera environment was also evaluated with several datasets, obtained from the original one by augmenting the number of records (from x2 to x10). It was find that the algorithm behaved linearly on the size of the dataset.

4 Differentially Private K-Means Clustering using Spark

In this section we describe the second solution we have developed to handle sensitive datasets, while protecting the privacy of individuals. In contrast to the approach described in Section 3, whose goal is to release *individual information* about each participant of the dataset in the form of *microdata*, the goal of the this approach is to release *aggregated information* about all participants as the result of some *data analysis*. These include simple

statistical analyses yielding e.g. an histogram or contingency table, as well as more complex data mining tasks such as clustering or classification.

To achieve this goal, we apply differential privacy techniques. In a nutshell, these techniques proceed by *modifying the algorithm* that implements the corresponding data analysis in order to "mask" the contribution of each individual. As such, it requires a separate and independent treatment for each data analysis task. To identify the most relevant data analysis task carried out by the Plan Ceibal, we held a meeting with its data scientists where they exposed some case studies, and came to the conclusion that the most profitable option was to develop a differentially private version of the k-means algorithm, one of the simplest and most commonplace algorithm for clustering purposes.

4.1 Preliminaries

Next, we review the prerequisites of our solution, namely k-means clustering and the differential privacy framework.

The *k*-means|| Clustering Algorithm

Given a collection of data points in the d-dimensional Euclidean space, the goal of the k-means problem is to partition the data points into k sets, i.e. clusters, each of them identified by its center (i.e. means of the points in the cluster), such that the average (squared) distance of a data point to the nearest center is minimized. Figure 1 illustrates the problem over a 2-dimensional dataset with $k = 3$ clusters.

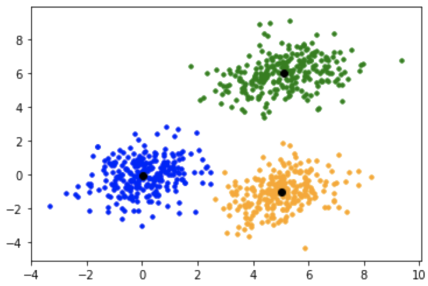


Figure 1: K-means clustering.

Since finding the optimal solution to the k-means problem is NP-hard [3], in practice one uses a heuristic to find approximate solutions. The heuristic, known as *Lloyd's Algorithm*, chooses k centers at random and iteratively refines them. In each iteration it i) assigns each data point to the cluster with the nearest center, and ii) updates the centers of the resulting clusters. The algorithm halts when the solution stabilizes, e.g. when between two consecutive iterations the centers move less than a given threshold distance or the points that change from one cluster to another are below a given threshold fraction.

Even though appealing, this heuristic has two major drawbacks. First, the running time

can be exponential in the worst case, and second, it yields a solution that can be arbitrarily far away from the optimal solution. To address the second issue, Arthur and Vassilvitskii [1] proposed a more informed initialization step: Instead of choosing the initial k centers at random, the idea is to spread them out, considering the distribution of data points that are being clustered. Concretely, the first center is chosen at random among the data points and each of the remaining center is chosen among the remaining data points, with probability proportional to the distance of the data point to the closest existing, i.e. already selected, center. As a result, data points further away from the already selected centers are more likely to be chosen as centers.

Even though this modification solves the accuracy problem of the vanilla k-means algorithm —it guarantees a bounded approximation ratio in $O(\log k)$—, it introduces a new problem: the algorithm initialization becomes inherently sequential since the probability with which a data point is chosen to be the i-th center critically depends on the selection of all previous $i-1$ centers. As a result, its applicability to massive data —a key requirement of our solution— is severely limited.[1]

Bahmani et al. [2] devised a solution for this limitation. Instead of doing k iterations, where in each iteration 1 center is incorporated, the proposed solution, dubbed k-means||, does only "a few" iterations, where in each iteration $\ell \in \Theta(k)$ (e.g. $2k$) centers are incorporated (in expectation). To this end, each data point is selected as a center independently, with a probability, as before, proportional to the distance to its closest existing center.[2]

Observe, however, that this process outcomes a set C of more than k centers (in expectation), say k'. To select exactly k centers as required, the k-means|| proceeds by assigning a weight w_c to each point $c \in C$, defined as the number of points that would be associated to the cluster with center c if we were to cluster the original dataset according to the centers in C. Finally, the set C is partitioned into k clusters considering the weighted of its points, which yields the final k centers that are to be used as initialization of the algorithm over the dataset. In practice, C is significantly smaller than original dataset, which allows this reclustering process to be done efficiently.

Apache Spark includes a distributed implementation of the above described k-means|| algorithm in its machine learning library MLlib. This fully-functional and optimized implementation of the algorithm is a central ingredient of our final solution, as described in Section 4.2.

Differential privacy

Intuitively, a randomized algorithm is differentially private if it behaves similarly on similar input datasets. To formalize this intuition, the framework of differential privacy relies on the notion of dataset adjacency: two datasets d, d' are called *adjacent*, written $d \sim d'$ iff they differ in a single data point.

Definition 4.1 (Differential privacy) *Given $\epsilon \geq 0$, we say that a randomized algorithm $\mathcal{A}$ is*

[1] Note that this modification in effect turns the center initialization into the *bottle-neck* of the algorithm, since the center refinements can already be parallelized

[2] Now, the probability normalization factor is accommodated so that ℓ centers are selected in expectation.

ϵ-differentially private *if for every pair of adyacent datapoints* d, d' *and every set* $S \subseteq \mathsf{range}(\mathcal{A})$*,*

$$\mathsf{Pr}\left[\mathcal{A}(d) \in S\right] \;\leq\; e^{\epsilon}\,\mathsf{Pr}\left[\mathcal{A}(d') \in S\right] \;.$$

Establishing differential privacy for numeric queries of limited sensitivity is relatively simple. The Laplacian mechanisms [4] says that we can obtain a differentially private version of query $f\colon D \to \mathbb{R}$ by simply perturbing its output: On input d, we return $f(d)$ plus some noise sampled from a Laplacian distribution. The noise must be calibrated according to the *global sensitivity* GS_f of f, which measures its maximum variation upon adjacent datasets; formally, $\mathsf{GS}_f = \max_{d,d'|d\sim d'} |f(d) - f(d')|$.

Theorem 4.1 (Laplacian mechanism) *Given a numeric query* $f\colon D \to \mathbb{R}$ *of global sensitivity* GS_f*, the randomized algorithm* $\mathcal{A}(d) \;=\; f(d) + \mathsf{Lap}\left(\frac{\mathsf{GS}_f}{\epsilon}\right)$ *is an* ϵ*-differentially private version of* f*.*

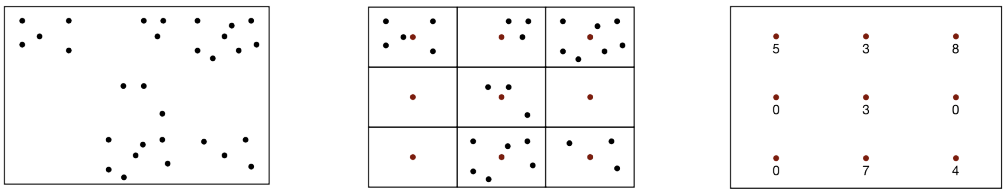
An appealing property fo

4.2 Approach description

In contrast to the traditional, so-called *interactive* approach, where one would modify the k-means algorithm itself to yield differentially private version, we adopt a *non-interactive* approach that exploits the post-processing property of differential privacy. Concretely, given a dataset we construct a differentially private *synopsis* or *abstraction* of the dataset, and then apply the k-means algorithm to the synopsis. The benefits of this approach are twofold. First, we can use Spark (or any other) implementation of k-means|| as is, taking advantage of all its carefully crafted optimizations and reducing the developing time. Second, we can use the differentially private synopsis not only for clustering purposes but also other data analysis tasks.

For constructing the dataset synopsis, we build on Qardaji et al. technique [5] originally designed for 2-dimensional datasets, and generalize it to d dimensions following the EUGkM approach of Su et al. [6, 7]. More specifically, we build the dataset synopsis adaptively, following a 2-levels partitioning approach. We begin by partitioning the dataset domain using a uniform grid and counting the number of points that lay within each cell. For abstraction purposes, all the points within the same cell are mapped to the cell center. Therefore, the dataset synopsis consists of all the cell centers together with the number of points that each center abstracts. An illustrative example is depicted in Figure 2. As we can see in the figure, for building the gird, each dimension is split into the same number of intervals (3, in this case), and in each dimension all intervals have the same width.

The so-obtained synopsis presents, however, two problems. First, it is not privacy-preserving. This can be fixed by releasing a noisy —rather than the exact— count of the number of points that each cell center abstracts, via the Laplace mechanism . Note that each point of the original datasets belongs to a single cell, and thus affects the count of a single center. In view of the parallel composition theorem of differential privacy , it is thus not necessary to *split* the available privacy budget among all applications of the Laplace



(a) Original dataset. (b) Dataset with grids and cell centers. (c) Dataset synopsis.

Figure 2: Dataset synopsis via uniform grids.

mechanism, but we can calibrate each application with the entire privacy budget, yielding this way more precise results.

The second limitation originates from the partition strategy, which completely disregards the datapoint distribution. As a result, there might be some cells with very few points and, at the same time, other cells with a lot of points, which results in a poor abstraction of the original dataset. Intuitively, we would expect a finer partitioning for dense regions, and a coarser partitioning for regions with few points, which is particularly desirable in the presence of sparse datasets. To address this phenomenon and produce more precise and efficient synopses, we employ a *two levels* abstraction, where we first lay a coarse grid and then further refine each of the 1st-level cells, according to their noisy counts: the more populated the cells are, the finer the grid we use to refine them. An illustrative example of this *two levels* abstraction is depicted in Figure 3.

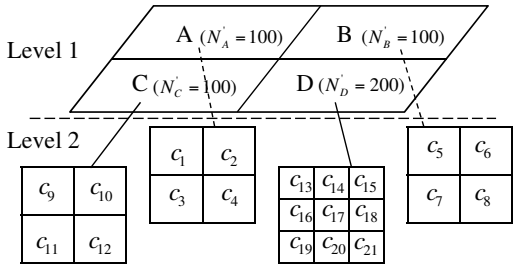


Figure 3: Adaptive, 2-levels abstraction via uniform grids.
Source: [9].

The final synopsis consists in the centers of the 2nd-level cells, together with the number of points from the original dataset that each of them abstracts. As before, these counts must be perturbed using the Laplacian mechanism to ensure differential privacy.

To complete the description of the synopsis construction, we are only left to specify the privacy budget and partitioning granularity employed in each abstraction level. As for the first point, if ϵ is the overall available privacy budget, a fraction $\alpha\epsilon$ is spent for perturbing the 1st-level counts, and the remaining $(1-\alpha)\epsilon$ is spent for perturbing the 2nd-level counts. For concreteness, we set $\alpha = 0.5$ but in practice we observe that the value of α has little impact on the final synopsis.

As for the second point, i.e. the partition granularity, assume we have a dataset of N

d-dimensional points. Then, the 1st-level abstraction partitions the data points domain into

$$M_1 = \left(\frac{N\epsilon}{c_1}\right)^{\frac{2d}{2+d}}$$

cells, by splitting each dimension into

$$m_1 = \left\lceil \max\left\{\sqrt[d]{100},\ \tfrac{1}{4}\sqrt[d]{M_1}\right\}\right\rceil$$

intervals of equal width . As for the 2nd-level abstraction, if N' is the noisy count associated to a 1st-level cell (if N' happens to be negative due to applied perturbation, it is zeroed), it is further partitioned into approximately

$$M_2 = \left(\frac{N'(1-\alpha)\epsilon}{c_2}\right)^{\frac{2d}{2+d}}$$

cells, by splitting each dimension into

$$m_2 = \left\lceil \sqrt[d]{M_2}\right\rceil$$

intervals of equal width.

References

[1] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035. Society for Industrial and Applied Mathematics, 2007. ISBN 9780898716245.

[2] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. *Proc. VLDB Endow.*, 5(7):622–633, Mar. 2012. doi: 10.14778/2180912.2180915.

[3] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine learning*, 56(1):9–33, 2004.

[4] C. Dwork. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer Verlag, July 2006. ISBN 3-540-35907-9. URL `https://www.microsoft.com/en-us/research/publication/differential-privacy/`.

[5] W. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 757–768, 2013. doi: 10.1109/ICDE.2013.6544872.

[6] D. Su, J. Cao, N. Li, E. Bertino, and H. Jin. Differentially private k-means clustering. In *Proceedings of the sixth ACM conference on data and application security and privacy*, pages 26–37, 2016.

[7] D. Su, J. Cao, N. Li, E. Bertino, M. Lyu, and H. Jin. Differentially private k-means clustering and a hybrid approach to private optimization. *ACM Transactions on Privacy and Security (TOPS)*, 20(4):1–33, 2017.

[8] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, 10(5):557–570, 2002. ISSN 02184885. doi: 10.1142/S0218488502001648.

[9] H. To, G. Ghinita, and C. Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *Proc. VLDB Endow.*, 7(10):919–930, June 2014. doi: 10.14778/2732951.2732966.