# Privacy-preserving Learning by Averaging in Collaborative Networks

Arijus Pleska


École Doctorale Mathematiques, Sciences du Numérique et de leurs Interactions (MADIS)

Thèse de Doctorat

# Privacy-preserving Learning by Averaging in Collaborative Networks

Calcul des Moyennes dans des Réseaux Collaboratifs pour l'Apprentissage Automatique et Préservant la Confidentialité

Préparée au sein de l'équipe Magnet, du laboratoire CRIStAL
et du centre de recherche INRIA Lille par

Arijus Pleska

sous la direction de

Jan Ramon

pour obtenir le grade de

Docteur en Informatique

Soutenue à **Villeneuve d'Ascq** le **6 juin 2023** devant le jury composé de

| | | |
|---|---|---|
| Benjamin Nguyen | Professeur, INSA Centre Val de Loire | Rapporteur |
| Martine De Cock | Professeur, University of Washington Tacoma | Rapportrice |
| Sébastien Gambs | Professeur, Université du Québec à Montréal | Examinateur |
| Romain Rouvoy | Professeur, Université de Lille | Président du jury |
| Jan Ramon | Directeur de Recherche, INRIA Lille | Directeur |

# Acknowledgements

I am thankful to Jan Ramon for the guidance throughout the work on this thesis.

I am thankful to Marc Tommasi, Mikaela Keller, and Aurélien Bellet for their comments on several extracts of this dissertation.

I am thankful to Benjamin Nguyen, Martine De Cock, Sébastien Gambs, and Romain Rouvoy for the participation in the defense of this thesis.

Also, I am thankful to Moitree Basu for her collaboration on the project on the tailored noise mechanism.

Likewise, I am thankful to César Sabater for the technical discussions and his comments on the context of this dissertation.

I am thankful to Mathieu Dehouck, Paul Mangold, and Nathalie Vauquier for their comments on several extracts of this dissertation and their help in the French translations.

I am thankful to Antoine Boutet for his welcome in presenting the project on distributed averaging without handshakes to the Privatics team; to the organizers of the 4-th MaDICS symposium for the opportunity to present in their event; to Michaël Perrot for his assistance in presenting my work to the Magnet team; and to Sylvain Salvati for the opportunity to exchange with students during la journée RIC (the day of research, innovation, and creativity) in 2021.

I am thankful to Mohamed Maouche, Rémi Gilleron, Pascal Denis, Damien Sileo, Brij Mohan Lal Srivastava, Mahsa Asadi, Gaurav Maheshwari, Mariana Vargas Vieyra, Onkar Pandit, Carlos Jorge Zubiaga Peña, William de Vazelhes, Kamal Macwan, Vitalii Emelianov, and Batiste Le Bars for their questions and insights.

I am thankful to Julie Jonas and Aurore Dalle for administrative assistance.

I am thankful to my dear friends for support.

# Abstract

In recent years, due to the growing importance of network applications and the growing concerns for privacy, there is an increasing interest in decentralized forms of machine learning. In this dissertation, we study the setting that involves a communication network of agents, where each agent locally privatizes (adds noise to) its data, and where the agents aim to collaboratively learn statistical models over their data. Such local privatization is in line with a standard of data privacy known as local differential privacy, and local differential privacy is useful when alternatives, such as secure multi-party computation or central differential privacy performed by a trusted curator, are infeasible. However, local differential privacy results, typically, in worse utility (less accurate statistical models) compared to central differential privacy because, for the same privacy budget, local differential privacy adds more privatization noise than central differential privacy. The principal question of this dissertation is the following: given that local differential privacy must be used, how could the agents maximize the utility they achieve? We study two cases to address the stated principal question.

In the first case, we consider the problem of distributed averaging, where each agent intends to collaboratively compute the unbiased average over the individual values of all agents without revealing neither their sensitive attributes nor their degree (number of neighbors). Usually, existing works solve this problem by assuming that either (i) each agent reveals its degree to its neighbors or (ii) every two neighboring agents can perform handshakes (requests that rely on replies) in every exchange of information. Since such assumptions are not always desirable, we propose an approach that is handshake-free and where the degrees are privatized. In particular, we use a gossip algorithm that computes averages that are biased when the graph of agents is non-regular (when the vertices have unequal degrees) and then perform a procedure combining multiple biased averages for bias correction. We apply the proposed approach for fitting linear regression models. We prove the asymptotic guarantee that the mean squared error between the average of privatized attributes computed by our approach and the average of sensitive attributes is $\mathcal{O}\left(\frac{1}{n}\right)$, where $n$ is the number of agents.

In the second case, we consider a group of agents, where features (for fitting regression models) are computed by transforming sensitive attributes, and where the transformations have high-magnitude gradients or singularities. In such setting, there is a risk to magnify the privatization noise if the perturbed data is in an interval where the feature function has high-magnitude gradients. We provide a tailored noise mechanism for privatizing features by solving a convex program in such a way that (i) only pertinent intervals of transformations are selected, (ii) the

variance of privatization noise is minimized, and (iii) the biasedness of privatization noise is minimized.

# Résumé

Ces dernières années, les applications en ligne se sont beaucoup développées. Cela a attiré une plus grande attention sur les problèmes de confidentialité des données et motivé la recherche sur les formes décentralisées d'apprentissage automatique. Dans cette thèse, nous nous intéressons à la situation où les agents d'un réseau de communication souhaitent apprendre un modèle statistique de façon collaborative, tout en préservant la confidentialité de leurs données personnelles. Une façon de protéger ces données est de les obfusquer (bruiter) avant de les partager. Ce genre d'obfuscation locale est conforme à la confidentialité différentielle locale (un standard d'obfuscation des données), et la confidentialité différentielle locale est utile lorsque d'autres solutions, reposant sur le calcul multipartite sécurisé ou sur la confidentialité différentielle centrale realisée par un tiers de confiance jouant le rôle d'orchestrateur, sont irréalisables. Cependant, la confidentialité différentielle locale souffre généralement d'une utilité moindre (les modèles statistiques sont moins précis) que la confidentialité différentielle centrale car, pour le même budget de confidentialité, la confidentialité différentielle locale doit ajouter plus de bruit que la confidentialité différentielle centrale pour obfusquer les données. La question principale de cette thèse est la suivante : en garantissant la forme locale de la confidentialité différentielle, comment les agents peuvent-ils maximiser l'utilité qu'ils obtiennent ? Nous répondons à cette question dans deux cas particuliers.

Dans le premier cas, nous considérons le problème du calcul distribué, où les agents souhaitent estimer de façon collaborative la moyenne non-biaisée de l'ensemble des valeurs individuelles de tous les agents, sans révéler ni leurs attributs sensibles ni leur degré (le degré d'un sommet étant le nombre de ses voisins). Généralement, les travaux existants résolvent ce problème en supposant soit (i) que les agents révèlent leur degré à leurs voisins respectifs, soit (ii) que toutes les paires de voisins peuvent éffectuer des handshakes (pour s'assurer de la réponse de chacun). Puisque de telles hypothèses ne sont pas toujours réalisables, nous proposons une approche qui ne nécessite pas de handshakes et qui ajoute du bruit aux degrés. En particulier, nous utilisons un algorithme de bavardage qui calcule des moyennes biaisées quand le graphe est non-régulier (quand tous les sommets n'ont pas le même degré), puis nous appliquons une procédure combinant les moyennes biaisées pour en corriger le biais. Nous appliquons ensuite l'approche proposée pour estimer des modèles de régression linéaire. Nous prouvons que, asymptotiquement, l'erreur quadratique moyenne entre la moyenne des attributs cachés (par le bruit) calculée par notre approche et la véritable moyenne des attributs sensibles est $\mathcal{O}\left(\frac{1}{n}\right)$, où $n$ est le nombre d'agents.

Dans le second cas, nous considérons un groupe d'agents, où les features (valeurs

entrant dans l'estimation des modèles de régression linéaire) sont calculées par application de fonctions sur des attributs sensibles, et ces fonctions présentent une grande amplitude de gradient ou des singularités. Dans une telle situation, il existe un risque d'amplifier le bruit d'obfuscation si les données perturbées se trouvent dans un intervalle où ladite fonction a une grande amplitude de gradient. Nous proposons un mécanisme de bruitage spécifique qui cache les features en résolvant un problème d'optimisation de telle sorte que (i) seuls des intervalles pertinents pour les fonctions considérées soient sélectionnés, (ii) la variance du bruit soit minimisée et (iii) le biais du bruit soit minimisé.

# Contents

# List of Figures

# List of Algorithms

# List of Tables

# Chapter 1

# Introduction

In recent years, due to the growing importance of network applications and the growing concerns for privacy, there is an increasing interest in decentralized forms of machine learning [NTC17; Col+16; VBT17].

On the one hand, reluctance to share sensitive data reduces the risk of unwanted profiling which can expose to targeted advertisement or discrimination in job applications. On the other hand, sharing such data can be useful in improving statistical models used in healthcare [Ter+22; Lam+21] or software maintenance [EPK14]. Data obfuscation is a computationally light approach to share data while preserving privacy with or without centralization. Upon centralization, an authority is entrusted to preserve privacy of each record shared with it. Upon decentralization, software on a user's device preserves privacy while records are shared in a network of users. When software is respectful towards regulation enhancing privacy, such as the General Data Protection Regulation, decentralization allows for preserving privacy without trusting an authority [NH21]. However, decentralization requires additional care, such as failure detection or tailored design of communication protocols.

We do not consider cryptographic approaches, such as secure multi-party computation [Sot+21], mainly due to additional computational cost. Also, we aim for studying networks where information flow is characterized by absence of handshakes (requests that rely on replies). As in such case, interaction among users is characterized by a stronger self-management because the users are not obliged to wait for each other.

In this dissertation, we study the setting that involves a communication network of agents, where an agent can communicate to another agent only if there is a communication link between them (we refer to such agents as neighboring agents), where each agent is attributed a vector of individual values, and where each agent intends to collaboratively compute statistics over the individual values of all agents for fitting statistical models. Furthermore, we consider that every agent intends to keep its individual values private (i.e., individual values are sensitive attributes). We recall two common approaches for computing statistics over the individual values of all agents in a privacy-preserving way:

- In the centralized approach, the community of agents chooses a trust-worthy agent (a trusted central curator). Then, each agent shares its sensitive

attributes with the central curator, the central curator computes statistics over the sensitive attributes of all agents, and the central curator privatizes the statistics. At this point, the central curator can share the privatized statistics with the agents.

- In the decentralized approach, each agent privatizes its sensitive attributes locally and shares the privatized attributes with its neighboring agents. Then, the community of agents commits to a procedure where each agent iteratively performs computations over the received values and shares the results with the neighbors. As the number of iterations increases, such information dissemination can be tailored to lead to statistics of interest over the privatized attributes.

First, we discuss the centralized approach in more detail. We assume that a central curator is connected to every agent (otherwise, the community adds communication links so that the central curator is connected to every agent). This way, each agent can share its sensitive attributes with the central curator, the central curator computes and then privatizes (adds appropriate amount of noise to) the statistics of interest, and, finally, the central curator shares the privatized statistics with the community.

Next, we discuss the decentralized approach in more detail. Even though no central curator is selected, the community can compute statistics from locally privatized attributes using a distributed algorithm. A suitable choice of such distributed algorithm is an iterative algorithm known as a gossip algorithm. In the initialization of a gossip algorithm, each agent shares its vector of individual values with its neighbors. In the first step, each agent computes the average from the receivable values. In the second step, each agent updates its shared value by the computed average. Under conditions studied later in this dissertation, iteration over the two steps leads to convergence, and each agent obtains the statistics of interest over the privatized attributes.

The statistics computed by the centralized approach are more accurate (which leads to a higher utility) than the statistics computed by the decentralized approach because the amount of noise to guarantee differential privacy (a standard of data privacy) is, typically, higher in a statistic computed from locally privatized attributes as opposed to a privatized statistic computed from sensitive attributes. However, the centralized approach relies on a central curator for privatization, whereas the decentralized approach does not. Meanwhile, the decentralized approach is susceptible to issues of distributed communication, e.g., inactivity of agents or the bias due to non-regular connectivity (when the graph of agents is non-regular, i.e., the numbers of neighbors vary).

In this dissertation, we reuse elements from the two aforementioned approaches. As in the centralized approach, we have a central curator. However, the role of the central curator is other than privatizing sensitive attributes. For example, such role can be solving mathematical optimization problems or averaging. As in the decentralized approach, each agent privatizes its sensitive attributes locally.

We focus on statistical models which can be fitted using statistics that are averages (later, we show that this is the case for linear regression). This way, we

are interested in distributed averaging of locally privatized attributes, which is an actively studied problem [DBR18; Bel+20]. Regarding real-world applications, distributed averaging with a minimal reliance on a central curator (which, typically, is an entity that provides or hosts an instance of software) is an element of self-managing distributed systems. In particular, a reduced reliance on a central curator reduces risks associated with robustness, infrastructure for data storage, and trust. For example, an approach relying on a central curator is susceptible to power cuts or information leakage. However, the design of a self-managing distributed system that functions well in practice requires consideration of numerous factors, such as inactive or malicious agents, convergence, asynchronicity, or the aforementioned bias due to non-regular connectivity.

We proceed by stating our principal question:

- Given that local differential privacy must be used, how could the agents maximize the utility they achieve? More precisely, how could the agents of a communication network maximize the accuracy of collaboratively computed averages of locally privatized attributes?

In this dissertation, we answer the latter question by the two following contributions.

In Section 3.3, we prove an asymptotic guarantee for the mean squared error between the average of sensitive attributes and the average of locally privatized attributes computed by a bias-correcting gossip algorithm, when the graph of agents is modeled by a random graph model parametrized by an arbitrary degree sequence, when the degrees (numbers of neighbors) of agents are sensitive, and when the agents interact without handshakes. We apply the proposed approach for fitting linear regression models while keeping the degrees private. We show on a synthetic graph dataset and real graph datasets that, when the features are polynomials of degrees, the regression model fitted by our approach can outperform the solution when locally privatized attributes are averaged by centralized averaging, i.e., the averaging function $f(\mathbf{x}) = \frac{1}{n}\sum_{i\in[n]} x_i$, where $\mathbf{x} \in \mathbb{R}^n$.

In Chapter 4, we provide a utility-maximizing mechanism for privatizing features that are computed by transforming sensitive attributes, when the transformations have high-magnitude gradients or singularities. In such setting, there is a risk of obtaining an outlier feature due to transforming a privatized attribute in an interval where the transformation has high-magnitude gradients. We mitigate the aforementioned risk by providing a tailored noise mechanism for privatizing features by solving a convex program in such a way that (i) only informative intervals of transformations are selected, (ii) the variance of privatization noise is minimized, and (iii) the biasedness of privatization noise is minimized. We show on synthetic datasets and a synthetically-extended real dataset that the tailored noise mechanism results in a higher utility compared to a classic privatization mechanism, when fitting a linear regression model where the feature function is either the logarithm, the reciprocal, or the tangent.

**Outline.** In Chapter 2, we discuss the background of the fields on which this dissertation relies. In Chapter 3, we study the bias in distributed averaging when the graph of agents is non-regular. In Chapter 4, we study the tailored

noise mechanism. In Chapter 5, we summarize this dissertation and discuss future directions.

# Chapter 2

# Background

In this chapter, we discuss the background of the fields on which this dissertation relies (illustrated in Figure 2.5). In particular, we review definitions, provide notation, and sometimes provide descriptions and examples.

At the beginning of this chapter, we review some basic concepts of linear algebra and familiarize with the general notation followed throughout this dissertation. In particular, we review the spectral norm that is used in our example of distributed averaging on particular random graphs known as Erdős–Rényi graphs (discussed in Section 3.2). Then, in Section 2.1, we review probability theory. In Section 2.2, we review graph theory. In Section 2.3, we review distributed systems. Afterwards, in Section 2.4, we review machine learning. In Section 2.5, we review data privacy. In Section 2.6, we review mathematical optimization. Finally, in Section 2.7, we briefly summarize how the reviewed fields relate to the settings discussed in this dissertation.

For a review of linear algebra, we recommend and mostly follow the textbook by Horn and Johnson [HJ12]. In this dissertation, we limit ourselves to the space $\mathbb{R}$ of real numbers (as opposed to the space $\mathbb{C}$ of complex numbers).

We start by recalling definitions of a field and a vector space; we do so with the intention to review the basis for definitions of vectors and matrices which are fundamental in linear algebra.

A field (also known as a scalar field) is a set of scalars closed under addition and multiplication; we denote a field by $F$. An $n$-dimensional vector space $S$ (also known as a vector space) over a field $F$ is a set of $n$-tuples, where the set $S$ meets the following conditions:

- $S$ is closed under vector addition (which is associative and commutative). That is, for each $\mathbf{u}, \mathbf{v} \in S$, we have $\mathbf{u} + \mathbf{v} \in S$.
- $S$ includes an identity which is the vector of 0's; we denote the vector of 0's by $\mathbf{0}$. That is, $\mathbf{0} \in S$.
- $S$ includes the additive inverse of its each element. That is, for each $\mathbf{u} \in S$, we have $-\mathbf{u} \in S$, where $\mathbf{u} + (-\mathbf{u}) = \mathbf{0}$.
- $S$ is closed under scalar multiplication. That is, for each $c \in F$ and for each $\mathbf{u} \in S$, we have $c\mathbf{u} \in U$.

An element of a vector space over a field $F$ is known as a vector. A finite collection of vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$ is linearly dependent if and only if there are scalars $c_1, c_2, \ldots, c_k$, not all zero, such that

$$c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + \ldots + c_k\mathbf{u}_k = \mathbf{0},$$

where $k \in \mathbb{N}_1$, and $\mathbb{N}_1 = \mathbb{N} \setminus \{0\}$. Further, we remark that by $F^n$ we denote a set of $n$-tuples whose elements are in $F$. Similarly, by $F^{m \times n}$ we denote a set of $m \times n$ arrays whose elements are in $F$. An element of $F^{m \times n}$ is known as a matrix. Essentially, a vector is a collection of elements over one axis (e.g., a sequence), and a matrix is a collection elements over two axes (e.g., a table). This way, a vector can be also interpreted as a matrix whose second axis is of length 1.

We summarize the notation of matrices, vectors, and scalars, which we have chosen to follow throughout this dissertation. Let $\mathbf{x} \in \mathbb{R}^n$ be a vector and let $\mathbf{Y} \in \mathbb{R}^{n \times n}$ be a matrix. When we refer to a particular element of $\mathbf{x}$, we use an index in the subscript, e.g., $x_2$ denotes the 2-nd element of $\mathbf{x}$ (we remark that $x_2$ takes the italic font and $\mathbf{x}$ takes the roman bolded font). Similarly, $y_{1,2}$ denotes the element in the 1-st row and the 2-nd column of $\mathbf{Y}$ (we remark that $y_{1,2}$ takes the lower-case and $\mathbf{Y}$ takes the upper-case). This way, we usually denote scalars and vectors in lower-case and matrices in upper-case. When we refer to some particular row or column of a matrix, we use the colon symbol (i.e., ":"). For example, $\mathbf{Y}_{:,2}$ refers to the 2-nd column of $\mathbf{Y}$ (as an exception to the general notation conventions followed in this dissertation, we remark that the vector $\mathbf{Y}_{:,2}$ keeps the upper-case opposed to taking the lower-case).

In Table 2.1, we provide a summary of the general notation followed throughout the dissertation.

Table 2.1: General notation followed throughout the dissertation

| Notation | Meaning |
|---|---|
| $x$ | Italic lower-case usually denotes a scalar or a map |
| $\mathbf{x}$ | Roman bolded lower-case denotes a vector |
| $\mathbf{X}$ | Roman bolded upper-case usually denotes a matrix |
| $X$ | Italic upper-case usually denotes a random variable or a set |
| $\boldsymbol{X}$ | Italic bolded upper-case usually denotes a random matrix |
| $\mathbb{N}_1$ | $\mathbb{N} \setminus \{0\}$ |
| $[x]$ | $\{i \leq x : i \in \mathbb{N}_1\}$ $(x \in \mathbb{N}_1)$ |
| $\mathbf{1}$ | Vector of 1's |
| $\mathbf{0}$ | Vector of 0's |
| $\mathbf{I}_k$ | $k \times k$ identity matrix (1's in the main diagonal, 0's elsewhere) |

We proceed by recalling definitions of a square matrix and a diagonal matrix.

A square matrix is a matrix with an equal number of columns and rows. A diagonal matrix is a square matrix where non-zero elements are only in its main diagonal, and the main diagonal of an $n \times n$ matrix $\mathbf{A}$ is the diagonal (a vector) that contains the elements $a_{1,1}, a_{2,2}, \ldots, a_{n,n}$. This way, the first element of the

main diagonal is at the upper-left corner of $\mathbf{A}$ and the last element of the main diagonal is at the bottom-right corner of $\mathbf{A}$.

We proceed by recalling the following basic operations that can be performed on matrices: the transpose, matrix inversion, and eigendecomposition.

Let $\mathbf{A}$ be an $m \times n$ matrix. The transpose is the operation that provides the $n \times m$ matrix $\mathbf{A}^{\mathrm{T}}$ whose $(i, j)$-th element is $a_{j,i}$, for each $i \in [n]$ and $j \in [m]$. We denote the transpose by the suffix of $\mathbf{A}^{\mathrm{T}}$.

Let $\mathbf{A}$ be an $n \times n$ matrix. Matrix inversion is the operation that provides a matrix $\mathbf{A}^{-1}$ (if it exists) so that the following equality is met:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}_n,$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix. We denote matrix inversion by the suffix of $\mathbf{A}^{-1}$.

Before reviewing eigencomposition, we remark that an eigenvalue and an eigenvector of a matrix $\mathbf{A}$ are, respectively, a scalar $\lambda$ and a non-zero vector $\mathbf{u}$ which result in the following equality:

$$\mathbf{A}\mathbf{u} = \lambda\mathbf{u}.$$

Eigendecomposition of an $n \times n$ matrix $\mathbf{A}$ with $n$ linearly independent eigenvectors is the pair of a matrix $\mathbf{U}$ and a matrix $\mathbf{\Lambda}$, where the columns of $\mathbf{U}$ contain the eigenvectors of $\mathbf{A}$ and $\mathbf{\Lambda}$ is a diagonal matrix that contains the eigenvalues of $\mathbf{A}$ in the order that is respective to the eigenvector alignment in the columns of $\mathbf{U}$. Eigendecomposition of $\mathbf{A}$ can be expressed in factors as follows:

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}.$$

For the previous definition, we consulted the textbook by Johnson et al. [JRA02]

Finally, we recall definitions of a vector norm, a matrix norm, an induced matrix norm, the spectral norm, and then discuss a useful result between the spectral norm and the largest singular value, where a singular value is the square root of an eigenvalue.

Let $S$ denote a vector space over $\mathbb{R}$. A norm $\|\cdot\|$ is a function $\|\cdot\| : S \to \mathbb{R}$ which, for each $\mathbf{u}, \mathbf{v} \in S$, meets the following conditions:

- $\|\mathbf{u}\| \geq 0$.
- $\|\mathbf{u}\| = 0$ if and only if $\mathbf{u} = \mathbf{0}$.
- $\|c\mathbf{u}\| = |c| \, \|\mathbf{u}\|$, for each $c \in \mathbb{R}$.
- $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$.

Essentially, a norm is a function that assigns a length to a vector. We remark that the fourth of the previously stated conditions is known as the triangle inequality.

Common examples of norms are the $l_1$ norm and the $l_2$ norm.

Let $\mathbf{u} \in \mathbb{R}^n$. The $l_1$ norm $\|\cdot\|_1$ is defined as follows:

$$\|\mathbf{u}\|_1 = \sum_{i \in [n]} |u_i|.$$

In other words, the $l_1$ norm of a vector is the sum of the absolute values of the elements of the vector.

The $l_2$ norm $\|\cdot\|_2$ (also known as the Euclidean norm) is defined as follows:

$$\|\mathbf{u}\|_2 = \sqrt{\sum_{i\in[n]} u_i^2}.$$

In other words, the $l_2$ norm of a vector is the square root of the sum of the squares of the elements of the vector.

Let $S$ denote a set of $n \times n$ matrices over $\mathbb{R}$. A matrix norm $\|\cdot\|$ is a function $\|\cdot\| : S \to \mathbb{R}$, which, for each $\mathbf{A}, \mathbf{B} \in S$, meets the following conditions:

- $\|\mathbf{A}\| \geq 0$.
- $\|\mathbf{A}\| = 0$ if and only if all elements of $\mathbf{A}$ are 0's.
- $\|c\mathbf{A}\| = |c| \, \|\mathbf{A}\|$, for each $c \in \mathbb{R}$.
- $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$.
- $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \, \|\mathbf{B}\|$.

Essentially, a matrix norm is a function that assigns a size to a matrix.

Let $\mathbf{x} \in \mathbb{R}^n$. Let $\mathbf{Y} \in \mathbb{R}^{n\times n}$. A matrix norm $\|\cdot\|$ induced by a norm $\|\cdot\|$ on an $n$-dimensional vector space over $\mathbb{R}$ (also known as an induced matrix norm) is defined as follows:

$$\|\mathbf{Y}\| = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{Yx}\|}{\|\mathbf{x}\|}.$$

The spectral norm $\|\cdot\|_2$ is a matrix norm induced by the $l_2$ norm $\|\cdot\|_2$ on an $n$-dimensional vector space over $\mathbb{R}$.

We recall the following result showing a connection between the spectral norm of $\mathbf{Y}$ and the largest singular value of $\mathbf{Y}$:

$$\begin{aligned} \|\mathbf{Y}\|_2 &= \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{Yx}\|_2}{\|\mathbf{x}\|_2} \\ &= \sigma_1(\mathbf{Y}), \end{aligned} \tag{2.1}$$

where $\sigma_1(\mathbf{Y})$ denotes the largest singular value of $\mathbf{Y}$. We remark that the previous result is proved in Horn and Johnson [HJ12].

## 2.1 Probability theory

In this section, we review probability theory. In particular, we review some preliminaries of the following fields on which this dissertation relies: random graphs (discussed in Subsection 2.2.2), machine learning (discussed in Section 2.4), and data privacy (discussed in Section 2.5). Also, we review some techniques used in our example of distributed averaging on particular random graphs known as Erdős–Rényi graphs (discussed in Section 3.2) and in our contributions on distributed

averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4).

At the beginning of this section, we review some basic concepts of probability theory, such as random variables and sampling. In Subsection 2.1.1, we review estimation of some basic types of statistics, such as the mean and the variance. In Subsection 2.1.2, we review common probability distributions. Finally, in Subsection 2.1.3, we review statistical significance.

For a review of probability theory, we recommend and mostly follow the textbook by Wasserman [Was04]. For a supplementary reference to definitions of probability theory, we recommend the dictionary by Everitt and Skrondal [ES10].

We start by recalling definitions of the following basic concepts of probability theory: a trial, a sample space, and an event.

A trial (also known as a statistical experiment) is a procedure that provides an outcome from a set of possible outcomes. We remark that outcomes of trials may be used to model various real-world phenomena. The domain of outcomes of a trial is referred to as a sample space; we denote it by $\Omega$. An event is a subset of a sample space $\Omega$.

We proceed by recalling definitions of disjoint sets, a partition, a power set, and a $\sigma$-algebra. For a review of definitions related to sets, we recommend the textbook by Halmos [Hal60].

Disjoint sets are two distinct sets whose intersection is the empty set. In other words, disjoint sets have no elements in common. When we have a collection of sets, where any two sets in the collection are disjoint, then we can refer to such collection as a collection of pairwise disjoint sets.

Let $k \geq 2$. A partition of a set $A$ is a set of pairwise disjoint sets $B_1, B_2, \ldots, B_k \subset A$, so that $B_1 \cup B_2 \cup \ldots \cup B_k = A$. Similarly as in Szemerédi [Sze75], we refer to the pairwise disjoint sets $B_1, B_2, \ldots, B_k$ as partition classes.

The power set of a set $A$ is the set of all subsets of $A$; we denote it by $\mathcal{P}(A)$. A $\sigma$-algebra on a set $A$ is a non-empty set $\Sigma \subseteq \mathcal{P}(A)$ which meets the following conditions:

- The empty set $\emptyset \in \Sigma$.
- If $B_1, B_2, \ldots, B_k \in \Sigma$, then $B_1 \cup B_2, \cup \ldots \cup B_k \in \Sigma$.
- If $B \in \Sigma$, then the set difference $A \setminus B \in \Sigma$.

We proceed by recalling definitions of a probability measure and a probability, and then stating the addition law of probability.

Let $\mathcal{S}$ be a set of events, so that $\mathcal{S}$ is a $\sigma$-algebra on a sample space $\Omega$. A probability measure (also known as a probability distribution) is a function $p : \mathcal{S} \to [0, 1]$ that satisfies the following conditions:

- $p(B) \geq 0$, for each $B \in \mathcal{S}$.
- $p(\Omega) = 1$.
- $p(B_1 \cup B_2) = p(B_1) + p(B_2)$, for each two events $B_1, B_2 \in \mathcal{S}$ that are disjoint sets.

The value $p(B)$ is the probability of an event $B \in \mathcal{S}$. Essentially, the higher the probability of an event, the more likely it is that the event contains the outcome that occurs. For example, if $p(B) = 0$, then we know that no outcome in $B$ can occur.

The addition law of probability is the theorem which states that, for two events $B_1$ and $B_2$, we have

$$\Pr(B_1 \cup B_2) = \Pr(B_1) + \Pr(B_2) - \Pr(B_1 \cap B_2), \tag{2.2}$$

where $B_1 \cup B_2$ denotes that an outcome occurs either in the event $B_1$ or the event $B_2$, and $B_1 \cap B_2$ denotes that an outcome occurs in both events $B_1$ and $B_2$. For the previous theorem, we consulted the textbook by DeGroot and Schervish [DS12].

We proceed by recalling definitions of a measurable space and a measurable function. For a review of definitions related to measure theory, we recommend the textbook by Shao [Sha03].

A measurable space is a pair $(X, \Sigma)$ of a set $X$ and a $\sigma$-algebra on $X$. Let $(X_1, \Sigma_1)$ and $(X_2, \Sigma_2)$ be measurable spaces. A function $f : (X_1, \Sigma_1) \to (X_2, \Sigma_2)$ is a measurable function if and only if $f^{-1}(\Sigma_2) \subset \Sigma_1$. That is, for every set $X_2' \in \Sigma_2$, we have

$$\{x_1 \in X_1 : f(x_1) \in X_2'\} \in \Sigma_1.$$

Having stated previous definitions, we are ready to recall a definition of a random variable. A random variable is a measurable function $X : (\Omega, \mathcal{S}, p) \to (\mathbb{R}, \mathcal{B})$, where $\mathcal{B}$ is the smallest $\sigma$-algebra that contains all open subsets of $\mathbb{R}$ ($\mathcal{B}$ is known as a Borel $\sigma$-algebra), $\mathcal{S}$ is a set of events that is a $\sigma$-algebra on $\Omega$, the triple $(\Omega, \mathcal{S}, p)$ is known as a probability space, and the pairs $(\Omega, \mathcal{S})$ and $(\mathbb{R}, \mathcal{B})$ are measurable spaces.

We highlight that the probability measure $p$ assigns a measure (a scalar greater or equal to 0 and lower or equal to 1) to every outcome in the sample space $\Omega$. Also, we remark that when we use the expression "an outcome of a random variable", we refer to "an outcome of a trial that is modeled by a random variable". For conciseness, we sometimes shorthand the expression "$X : (\Omega, \mathcal{S}, p) \to (\mathbb{R}, \mathcal{B})$" by "$X : \Omega \to \mathbb{R}$".

When speaking of random variables, we either consider discrete random variables or continuous (real-valued) random variables. We remark that we use a broader definition of a discrete random variable mainly because later in this dissertation we consider random graphs.

We recall a definition of a discrete random variable. A discrete random variable is a function $X : \Omega \to \mathbb{R}$ that takes on either a finite number of values (i.e., $x_1, x_2, \ldots, x_n \in \Omega$, where $n \in \mathbb{N}_1$) or a countably infinite number of values (i.e., $x_1, x_2, \ldots \in \Omega$) according to a function $p_X(x) : \Omega \to [0, 1]$ known as the probability mass function and defined by

$$p_X(x) = \Pr(X = x),$$

where $\Pr(X = x)$ denotes the probability that an outcome of $X$ takes a value $x$, and where the probability mass function must satisfy the following conditions:

- $p_X(x) \geq 0$, for each $x \in \Omega$.

- $\sum_{x \in \Omega} p_X(x) = 1$.

We recall a definition of a continuous (real-valued) random variable. A continuous random variable is a function $X : \Omega \to \mathbb{R}$ that takes values in $\Omega$ according to a function $p_X(x) : \mathbb{R} \to \mathbb{R}$ known as the probability density function and defined by

$$\int_a^b p_X(x)dx = \Pr(a \leq X \leq b),$$

where $a$ and $b$ are any scalars under the condition that $a \leq b$, $\Pr(a \leq X \leq b)$ is the probability that an outcome of $X$ takes a value that is in the interval $[a, b]$, and where the probability density function must satisfy the following conditions:

- $p_X(x) \geq 0$, for each $x \in \mathbb{R}$.
- $\int_{-\infty}^{\infty} p_X(x)dx = 1$.

Essentially, probability mass functions and probability density functions can be interpreted as probability distributions. When a random variable $X$ follows a probability distribution $p$, this is sometimes denoted as follows:

$$X \sim p.$$

We remark that denoting the probability distribution by $p$ (instead of $p_X$) might reduce ambiguity when we have several random variables that follow the same probability distribution.

In Table 2.2, we provide a summary of the introduced notation of basic concepts of probability theory.

Table 2.2: Introduced notation of basic concepts of probability theory

| Notation | Meaning |
|---|---|
| $\Omega$ | Sample space |
| $\mathcal{S}$ | Set of events |
| $A, B$ | Events |
| $X, Y$ | Random variables |

We review some more definitions that are more closely related to random variables: the cumulative distribution function, the quantile function, the support, and independent random variables.

The cumulative distribution function of a (real-valued) random variable $X$ is the function $\phi_X : \mathbb{R} \to [0, 1]$ defined by

$$\phi_X(x) = \Pr(X \leq x),$$

where $\Pr(X \leq x)$ is the probability that an outcome of a random variable is less than or equal to a value $x$.

The quantile function of a (real-valued) random variable $X$ is the function $\phi_X^{-1}(q) : [0,1] \to \mathbb{R}$ that is the inverse of the cumulative distribution $\phi_X(x)$ of $X$, it is defined as follows:

$$\phi_X^{-1}(q) = \inf\{x : \phi_X(x) > q\}.$$

The support of a random variable is the subset of its sample space, where the subset contains the elements that are mapped by the probability mass/density function to probabilities greater than 0.

Independent random variables are two random variables where an outcome of one random variable does not influence the outcome of the other. For the previous definition, we consulted the textbook by Tijms [Tij07]. The independence between random variables $X$ and $Y$ are denoted by $X \perp Y$. When we have a collection $\{X_i\}_{i\in[n]}$ of random variables, where each two random variables $X, X' \in \{X_i\}_{i\in[n]}$ are such that $X \perp X'$, then we refer to $\{X_i\}_{i\in[n]}$ as a collection of pairwise independent random variables.

We proceed by recalling definitions of a joint probability distribution and a marginal probability distribution; we do so with the intention to show a link and a distinction between the two. A joint probability distribution is a probability distribution that provides probabilities for outcomes of several random variables. To consider a particular discrete case, the joint probability mass function of discrete random variables $X_1, X_2, \ldots, X_n : \Omega \to \mathbb{R}$ is the function $p_{X_1,X_2,\ldots,X_n} : \Omega^n \to [0,1]$, defined by

$$p_{X_1,X_2,\ldots,X_n}(x_1, x_2, \ldots, x_n) = \Pr(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n).$$

A marginal probability distribution is a probability distribution that provides probabilities for outcomes of one or several random variables that define a joint probability distribution $p_{X_1,X_2,\ldots,X_n}(x_1, x_2, \ldots, x_n)$. For example, when we have a joint probability distribution $p_{X,Y}(x, y)$ of random variables $X$ and $Y$, then one of its two marginal probability distributions is

$$p_X(x) = \sum_{y\in\Omega} p_{X,Y}(x, y),$$

and the second marginal distribution is $p_Y(y)$.

We relate a conditional probability to a joint probability distribution. A conditional probability is a probability that is parametrized by a fixed outcome of another random variable. For random variables $X$ and $Y$, the conditional probability of an outcome $x$ of a random variable $X$ given an outcome $y$ of a random variable $Y$ is defined as follows:

$$p_{X|Y}(x \,|\, y) = \frac{p_{X,Y}(x, y)}{p_Y(y)}, \tag{2.3}$$

where $p_Y(y) > 0$. We remark that $p_{X|Y}$ is called the conditional probability distribution of $X$ given $Y$.

We recall Bayes' theorem. Let $k \geq 2$ and let events $B_1, B_2, \ldots, B_k$ form a partition of a sample space $\Omega$, where, for each $i \in [k]$, we have $\Pr(B_i) > 0$. Let

an event $A \subseteq \Omega$, where $\Pr(A) > 0$. Bayes' theorem (also known as Bayes' rule) is stated as follows:

$$\Pr(B_i \mid A) = \frac{\Pr(A \mid B_i) \Pr(B_i)}{\sum_{j \in [k]} \Pr(A \mid B_j) \Pr(B_j)}.$$

Since the law of total probability states that

$$\sum_{j \in [k]} \Pr(A \mid B_j) \Pr(B_j) = \Pr(A),$$

the denominator in Bayes' theorem can be substituted with $\Pr(A)$. In such case, the expression of Bayes' theorem resembles the conditional probability defined in Equation 2.3.

Finally, we recall definitions of a statistical population, sampling, a stochastic process, and a statistical model.

A statistical population (also known as a population) is a finite or infinite set of elements. Sampling is a procedure that selects a subset of elements from a statistical population. We remark that a subset obtained by sampling is referred to as a sample, and an element of a sample is referred to as an observation (also known as a realization or as a sample point). For example, uniformly-random sampling is sampling that leads to every element of a population being observed with the same probability. We remark that a statistical population can be modeled by a collection of independent and identically distributed random variables (by identically distributed, we mean that the random variables follow the same probability distribution). This way, a set of outcomes of independent and identically distributed random variables can be interpreted as a sample of a statistical population.

A stochastic process is a finite or infinite sequence $(X_t)_{t \in T}$ of random variables, where $X_t$ is a random variable at time $t$, and $T$ is a time range. Typically, $T = \{0, 1, 2, \ldots\}$.

A statistical model is a pair of a sample space and a set of probability distributions on the sample space. For the previous definition, we consulted McCullagh [McC02]. For statistical models, it is common to assume that observations are obtained from an underlying probability distribution. We remark that collecting such observations helps in identifying probability distributions that are good approximations of the underlying probability distribution.

### 2.1.1 Statistics and their estimation

In this subsection, we review some common statistics as well as estimation techniques. In particular, we review Hoeffding's inequality that is used in our example of distributed averaging on particular random graphs known as Erdős–Rényi graphs (discussed in Section 3.2). Also, we review U-statistics that are used in our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4).

To start with, we highlight that by a statistic we refer to a numerical characteristic that is either an average (e.g., the average age of a demographic population) or a count (e.g., the number of members in a demographic population).

We proceed by recalling definitions of an estimate and an estimator.

An estimate of a parameter $\theta$ is the following value:

$$\hat{\theta} = f(x_1, x_2, \ldots, x_n),$$

where $\{x_1, x_2, \ldots, x_n\}$ is a collection of observations of independent and identically distributed random variables $X_1, X_2, \ldots, X_n$ whose probability distribution is parametrized by $\theta$, and where $f$ is a real-valued function. In other words, $\hat{\theta}$ is an observation of the following random variable:

$$\Theta = f(X_1, X_2, \ldots, X_n),$$

where $\Theta$ is known as an estimator. We remark that the caret symbol (i.e., "ˆ") indicates that a term is an estimate, e.g., an estimate $\hat{\theta}$ approximates a parameter $\theta$. Also, we remark that a probability distribution parametrized by $\theta$ can be unknown yet observable. In such case, we sometimes refer to $\theta$ as the true value of a parameter of an unknown probability distribution. For the previous definitions, we consulted the textbook by DeGroot and Schervish [DS12].

We recall definitions of some common statistics that are related to random variables: the expected value, the variance, and the standard deviation.

The expected value $\mathbb{E}[\cdot]$ (or the mean) of a (real-valued) discrete random variable $X$ taking values $x_1, x_2, \ldots$ is defined as follows:

$$\mathbb{E}[X] = \sum_{i \in \{1,2,\ldots\}} x_i p_X(x_i),$$

assuming that the sum is well-defined, and where $p_X$ is the probability mass function of $X$.

Similarly, when $X$ is a continuous random variable, the expected value is defined as follows:

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x p_X(x) dx,$$

assuming that the integral is well-defined, and where $p_X$ is the probability density function of $X$. We remark that the mean $\mathbb{E}[X]$ of a continuous random variable is well-defined when $\int_{-\infty}^{\infty} |x| p_X(x) dx < \infty$. We denote the mean of a random variable by $\mu$.

The variance of a (real-valued) discrete random variable $X$ taking values $x_1, x_2, \ldots$ is defined as follows:

$$\operatorname{var}(X) = \sum_{i \in \{1,2,\ldots\}} (x_i - \mu_x)^2 p_X(x_i),$$

assuming that the sum is well-defined, where $p_X$ is the probability mass function of $X$, and where $\mu_x$ is the mean of $X$.

Similarly, when $X$ is a continuous random variable, the variance is defined as follows:

$$\operatorname{var}(X) = \int_{-\infty}^{\infty} (x - \mu_x)^2 p_X(x) dx,$$

assuming that the integral is well-defined, and where $p_X$ is the probability density function of $X$. We denote the variance of a random variable by $\sigma^2$. The standard deviation of a random variable is the squared root of its variance; we denote it by $\sigma$.

Having stated the definitions of the mean and the variance, we recall a useful tool in probability theory, which is a particular concentration inequality known as Hoeffding's inequality, and where a concentration inequality is a method for obtaining a bound on how far an observation of a random variable deviates from some fixed value, by also bounding the probability of such observation. Let $t > 0$. Let $X_1, X_2, \ldots, X_n$ be independent random variables, each of which has the mean and the variance that are well-defined, and where the support of each random variable is the interval $[a, b]$ with $a \leq b$ (i.e., random variables $X_1, X_2, \ldots, X_n$ are bounded). Let $S_n = \sum_{i \in [n]} X_i$. Hoeffding's inequality [Hoe63] is defined as follows:

$$\Pr(S_n - \mathbb{E}[S_n] \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i \in [n]} (b-a)^2}\right).$$

In other words, Hoeffding's inequality is a concentration inequality that bounds the difference between the sum $S_n$ and the expected value of $S_n$.

We highlight that Hoeffding's inequality can be rearranged to bound the difference between the average of random variables and the expected value of that average, that is

$$\Pr\left(\frac{1}{n}S_n - \mathbb{E}\left[\frac{1}{n}S_n\right] \geq t\right) \leq \exp\left(-\frac{2n^2t^2}{\sum_{i \in [n]} (b-a)^2}\right). \tag{2.4}$$

We proceed by reviewing some common statistics that are computed from samples: the sample mean and the sample variance.

The sample mean is the average computed over the elements in a sample. Let a sample $\{x_1, x_2, \ldots, x_n\}$ be a collection of observations of $n$ independent and identically distributed (real-valued) random variables. The sample mean is defined as follows:

$$\bar{\mu}_x = \frac{1}{n} \sum_{i \in [n]} x_i.$$

Similarly, the unbiased sample variance is defined as follows:

$$\begin{aligned} \bar{\sigma}_x^2 &= \frac{n}{n-1} \frac{1}{n} \sum_{i \in [n]} (x_i - \bar{\mu}_x)^2 \\ &= \frac{1}{n-1} \sum_{i \in [n]} (x_i - \bar{\mu}_x)^2. \end{aligned}$$

The denominator $n - 1$ (as opposed to $n$) in the expression of the unbiased sample variance corrects the bias due to the use of the sample mean $\bar{\mu}_x$ as opposed to the true mean $\mu_x$, where a bias (also known as an estimator bias) is defined as the difference between the expected value of an estimator and the true value of the parameter that is estimated by the estimator. The aforementioned bias correction

due to the use of the sample mean $\bar{\mu}_x$ as opposed to the true mean $\mu_x$ is known as Bessel's correction. For the previous definitions, we consulted the textbook by Radziwill [Rad15].

We remark that the sample mean $\bar{\mu}_x$ and the unbiased sample variance $\bar{\sigma}_x^2$ are examples of unbiased estimates, where an unbiased estimate is an estimate without the estimator bias. We remark the bar symbol (i.e., "$\bar{}$") indicates that a term is an unbiased estimate.

We review a family of unbiased estimates known as U-statistics [Hoe48] which are defined below.

**Definition 1.** *Let $r \in \mathbb{N}_1$. Let $n \geq r$. Let $\{x_1, x_2, \ldots, x_n\}$ be a sample of observations of $n$ independent and identically distributed random variables. Let $\phi : \mathbb{R}^r \to \mathbb{R}$ be a symmetric function. A U-statistic with kernel $\phi$ of degree $r$ is defined as follows:*

$$\bar{u}_x = \binom{n}{r}^{-1} \sum_{i_1 < \ldots < i_r} \phi(x_{i_1}, x_{i_2}, \ldots, x_{i_r}),$$

*where the sum is over all $r$-combinations of the elements in the sample ($x_{i_1}, x_{i_2}, \ldots, x_{i_r}$ are $r$ distinct elements in the sample).*

We remark that the sample mean and the sample variance are common examples of U-statistics. Also, we remark that the letter "U" in the term "U-statistic" abbreviates "unbiased". For a more detailed description of U-statistics, we recommend the lecture notes by Ferguson [Fer03].

### 2.1.2 Common probability distributions

In this subsection, we review some common probability distributions. In particular, we review the Bernoulli distribution, the Binomial distribution, and the Poisson distribution as preliminaries for random graphs (discussed in Subsection 2.2.2). Then, we review the uniform distribution, mainly, as a preliminary for graph generation (discussed in Subsection 2.2.3). We review the Laplace distribution and the Gaussian distributions as preliminaries for data privacy (discussed in Section 2.5). Finally, we review Student's $t$-distribution as a preliminary for defining a particular confidence interval (discussed in Subsection 2.1.3). For the definitions in this subsection, we consult the dictionary by Everitt and Skrondal [ES10].

The Bernoulli distribution is a probability distribution of a discrete random variable $X$ with outcomes either 1 or 0, and whose probability mass function is defined as follows:

$$\Pr(X = 1 \,|\, p) = p,$$
$$\Pr(X = 0 \,|\, p) = 1 - p,$$

where $p$ is the probability of a Bernoulli trial which is commonly illustrated as a coin toss where "heads" appear with probability $p$ and "tails" appear with probability $1 - p$. In our case, we interpret "heads" as the value 1.

The Binomial distribution is a probability distribution of a discrete random variable $X$ whose probability mass function is defined as follows:

$$p_X(k \mid n, p) = \binom{n}{k} p^k (1-p)^{n-k},$$

where $p$ is the probability of a Bernoulli trial resulting in the value 1, $n$ is the number of independent Bernoulli trials, and $k$ is the number of Bernoulli trials that result in the value 1. We interpret the Binomial distribution as a probability distribution modeling the number $k$ of occurrences of "heads" (outcomes of the value 1) in a sequence of $n$ independent Bernoulli trials.

The Poisson distribution is a probability distribution of a discrete random variable $X$ whose probability mass function is defined as follows:

$$p_X(k \mid \lambda) = \frac{\lambda^k}{k!} e^{-\lambda},$$

where $\lambda > 0$ is a parameter known as the rate and $k \in \mathbb{N}$ is a parameter known as the count. We interpret the Poisson distribution as a probability distribution modeling occurrence of $k$ events in an interval of time, given that the events may occur at a fixed rate $\lambda$, independently of previous occurrences.

The uniform distribution is a probability distribution of a continuous random variable $X$ whose probability density function is defined as follows:

$$p_X(x \mid a, b) = \frac{1}{b-a},$$

where $a < b$. The uniform distribution is characterized by all its outcomes having the same probability. The support of a random variable that follows the uniform distribution is the interval $[a, b]$. We denote the uniform distribution by $\mathrm{uni}(a, b)$.

The Laplace distribution is a probability distribution of a continuous random variable $X$ whose probability density function is defined as follows:

$$p_X(x \mid \mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}},$$

where $\mu$ is a parameter known as the mean and $b > 0$. We denote the Laplace distribution by $\mathrm{lap}(\mu, b)$.

The Gaussian distribution (also known as the Normal distribution) is a probability distribution of a continuous random variable $X$ whose probability density function is defined as follows:

$$p_X(x \mid \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

where $\mu$ is a parameter known as the mean and $\sigma^2 > 0$ is a parameter known as the variance. We denote the Gaussian distribution by $\mathcal{N}(\mu, \sigma^2)$.

Student's $t$-distribution is a probability distribution of a continuous random variable $X$ whose probability density function is defined as follows:

$$p_X(x \mid \nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}},$$

where $\nu \in \mathbb{N}_1$ is a parameter known as the degrees of freedom and $\Gamma(\nu) = (\nu - 1)!$ is the gamma function. We remind that Student's $t$-distribution can be used to define a particular confidence interval for evaluating statistical significance (we review statistical significance in Subsection 2.1.3).

In Figure 2.1, we illustrate two probability density functions with the intention to indicate that the flatness of probability density functions can give a comparative indication of the variance of the corresponding probability distributions. In particular, a flatter probability density function indicates a higher variance compared to a steeper probability density function.

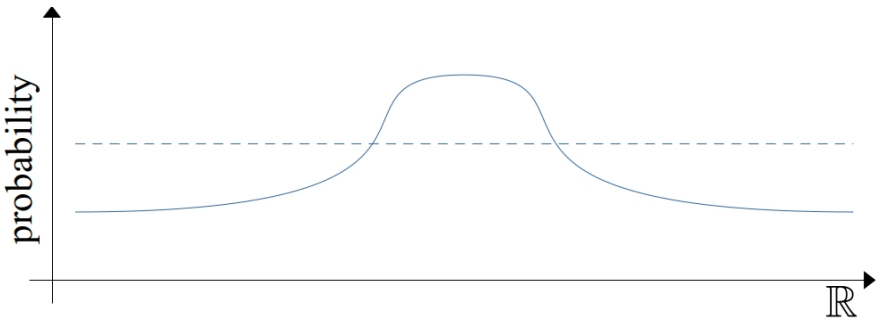


Figure 2.1: Illustration of two probability density functions (expressed as a dashed curve and a filled curve). The variance corresponding to the dashed curve is higher, since the filled curve has a region of higher probabilities.

### 2.1.3 Statistical significance

In this subsection, we elaborate on statistical significance. In particular, we review a confidence interval for a population mean, as such confidence interval is used in the experiments of our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4). For the definitions in this subsection, we consult the textbook by Weiss [Wei12].

We start by recalling definitions of the null hypothesis, the alternative hypothesis, statistical significance, $p$-value, and the significance level. The intention of this is to discuss an interpretation of the significance level.

The null hypothesis is defined as the statement that two statistical models are the same. In relation to that, the alternative hypothesis is defined as the statement that the two statistical model differ significantly. One accepts the alternative hypothesis by gathering evidence that (under statistical significance) allows for rejecting the corresponding null hypothesis.

Statistical significance is a measure of unlikeness of an event occurring under the assumption of the null hypothesis. The $p$-value is the probability of the event occurring under the assumption that the null hypothesis is true, it is sometimes denoted by $p$. The significance level is the probability of rejecting the null hypothesis that was assumed to be true; we denote it by $\alpha$. It is claimed that the event is statistically significant if $p \leq \alpha$. When speaking of statistical significance in the context of computer science research problems, it is common to fix the significance

level to $\alpha = 0.05$. This way, in this dissertation, we fix the significance level to $\alpha = 0.05$.

We proceed by recalling a definition of a confidence interval for the population mean.

A $(1 - \alpha)$-confidence interval (also known as a confidence interval) for the population mean is an interval estimate which contains the mean of a statistical population with the confidence level $1 - \alpha$.

We proceed by recalling a definition of the confidence interval based on Student's $t$-distribution. We remark that, for this, we follow a particular procedure known as the one-mean $t$-interval procedure.

The confidence interval that is based on Student's $t$-distribution is the following interval:

$$[\bar{\mu}_x - l, \bar{\mu}_x + l], \tag{2.5}$$

where $\bar{\mu}_x$ is the sample mean of a sample $\{x_1, x_2, \ldots, x_n\}$ with $n$ observations from a given statistical population,

$$l = \phi^{-1}\left(1 - \frac{\alpha_{\text{ci}}}{2} \mid \nu\right) \frac{\bar{\sigma}_x}{\sqrt{n}}$$

is the half-length of the confidence interval, $\bar{\sigma}_x$ is the square root of the unbiased sample variance of the sample, $\nu = n - 1$ are the degrees of freedom, $\alpha_{\text{ci}} = 0.05$ is the significance level, and $\phi^{-1}$ is the quantile function (the inverse of the cumulative distribution function $\phi$) of Student's $t$-distribution. We remark that the inversion of $\phi$ can result in a complex formula or the inverse might not have a closed-form expression. In such cases, $\phi^{-1}$ can be approximated using Taylor series, similarly as in Giner and Smyth [GS16]. Also, we remark that the middle point of the confidence interval in Equation 2.5 is the sample mean $\bar{\mu}_x$.

## 2.2 Graph theory

In this section, we review graph theory. In particular, we review some preliminaries related to execution of distributed algorithms on graphs (discussed in Subsection 2.3.2). Also, we review some preliminaries of random graphs models that are discussed in our example and contribution on the bias in distributed averaging when the graph of agents is non-regular (discussed, respectively, in Section 3.2 and Section 3.3).

At the beginning of this section, we review some basic concepts of graph theory. In Subsection 2.2.1, we precise to what we refer to when speaking of structural properties related to the edges of a graph. In Subsection 2.2.2, we review some common random graph models. In Subsection 2.2.3, we review some basics of graph generation.

For a review of graph theory, we recommend and mostly follow the textbook by Diestel [Die05]. For a supplementary reference to definitions of graph theory, we recommend the textbook by van Steen [Ste10].

We start by recalling definitions of the following basic concepts of graph theory: a graph, the order of a graph, the size of a graph, and a subgraph.

A graph is a pair $G = (V, E)$ of a set $V$ of vertices and a set $E$ of edges, where an element $v$ that belongs to $V$ is known as a vertex and an edge is a 2-element subset of $V$. Given vertices $v, u \in V$, we denote an edge between them either by $\{v, u\}$ or $vu$. We remark that graphs can be visualized as fixed points (vertices) that are joined by lines (edges).

The order of a graph $G = (V, E)$ is the number $|V|$ of vertices, and the size of the graph is the number $|E|$ of edges. The empty graph is the graph whose set of vertices and whose set of edges are empty sets.

A subgraph $G' = (V', E')$ of a graph $G = (V, E)$ is a graph whose vertices and edges are, respectively, a subset of $V$ and a subset of $E$. In other words, $V' \subseteq V$ and $E' \subseteq E$.

We proceed by recalling definitions of the following basic concepts that are related to edges: adjacent vertices, the degree of a vertex, the degree sequence, the degree distribution, an adjacency matrix, a walk, a path, a cycle, and the diameter of a graph.

Adjacent vertices (also known as neighboring vertices) are two vertices of a graph that have an edge in common. The neighborhood of a vertex $v$ is the set of all adjacent vertices of $v$; we denote the neighborhood of a vertex $v$ by $N(v)$.

The degree of a vertex is the number of edges incident with that vertex; we denote the degree of a vertex $v$ by $d_v$. In other words, the degree of a vertex is the cardinality of its neighborhood. The degree sequence of a graph is a sequence whose elements are the degrees of all vertices on that graph; we denote a degree sequence by $\mathbf{d}$. The degree distribution of a graph is the histogram (expressed in relative frequencies) of the degree sequence of the graph.

The adjacency matrix of a graph $G = (V, E)$ is the square $|V| \times |V|$ matrix $A$ whose $(i, j)$-th element $a_{i,j}$ is defined as follows:

$$a_{i,j} = \begin{cases} 1 & \text{when } v_i v_j \in E, \\ 0 & \text{otherwise.} \end{cases}$$

This way, $a_{i,j}$ indicates the presence (by 1) or the absence (by 0) of the edge between vertices $v_i, v_j \in V$.

A walk (of length $k$) in a graph $G$ is a sequence $(v_0, e_1, v_1, e_2, \ldots, v_{k-1}, e_k, v_k)$ that alternates between a vertex and an edge in such a way that, for each $i \in [k]$, we have that the edge $e_i = v_{i-1}v_i$. A closed walk is a walk where $v_0 = v_k$. A trail is a walk in which all edges are distinct. A path is a trail in which all vertices are distinct. A cycle (of length $k \geq 3$) is a trail that is a closed walk and in which all vertices except $v_0$ and $v_k$ are distinct. An acyclic graph (also known as a forest) is a graph without cycles. The diameter of a graph is the length of the shortest path between two vertices in the graph, where the size of such shortest path is higher or equal to the shortest path between any two vertices of the graph (in case a path between some two vertices does not exist, then the diameter is $\infty$).

We recall a definition of the edge density of the pair of two (non-empty) disjoint subsets of vertices on a graph $G = (V, E)$.

Let $E(V')$ denote the set of edges, where each edge in the set is incident with at least one vertex in a subset $V' \subseteq V$. Let $C, C' \subseteq V$ be (non-empty) disjoint

subsets. We define the edge density of the pair $(C, C')$ as follows:

$$\operatorname{den}(C, C') = \frac{|E(C \cap C')|}{|C|\,|C'|}. \tag{2.6}$$

In Table 2.3, we provide a summary of the introduced notation of basic concepts of graph theory.

Table 2.3: Introduced notation of basic concepts of graph theory

| Notation | Meaning |
|---|---|
| $G$ | Graph |
| $V$ | Set of vertices of $G$ |
| $E$ | Set of edges of $G$ |
| $v$ | Vertex in $V$ |
| $e$ | Edge in $E$ |
| $n$ | (Commonly denotes) the order of $G$ |
| $\deg(v), d_v$ | Degree of $v$ |
| $\mathbf{d}$ | Degree sequence $(d_1, d_2, \ldots, d_n)$ |
| $N(v)$ | Neighborhood of $v$ |

We proceed by recalling definitions of a connected graph, an (un)directed graph, a labeled graph, a weighted graph, a regular graph, a complete graph, a tree, a spanning subgraph, and a spanning tree.

A connected graph is a (non-empty) graph whose any two vertices can be linked by a path in the graph. A triangle graph (also known as a triangle) is a connected graph with 3 vertices.

A directed graph is a pair $G = (V, E)$ of disjoint sets of vertices and edges together with two maps $f_{\text{beg}} : E \to V$ and $f_{\text{end}} : E \to V$, where, for each edge $e \in E$, the maps assign, respectively, an initial vertex $f_{\text{beg}}(e) \in e$ and a terminal vertex $f_{\text{end}}(e) \in e$. When $f_{\text{beg}}(e) = f_{\text{end}}(e)$, the edge $e$ is known as a loop (also known as a self-loop). We remark that an undirected graph refers to a graph as defined at the beginning of this section.

Having the maps $f_{\text{beg}}$ and $f_{\text{end}}$, we can interpret a directed graph as a graph whose edges are ordered pairs of vertices, and the direction of an edge goes from the initial vertex to the terminal vertex. The edges of a directed graph can be visualized by curves with an arrow at one (or both) end-points, as opposed to an undirected graph, where curves are without added symbols at the end-points. This way, the arrow of a curve (an edge) of a directed graph indicates the direction of that edge (though we might also have an arrow at each end of the curve). We remark that the direction of an edge is important when defining a walk on a graph. When we have an edge between vertices $v$ and $u$, which corresponds to an unordered pair $(v, u)$, then a walk on the graph can move from $v$ to $u$ but not from $u$ to $v$.

A labeled graph is a graph where either every vertex, edge, or both are associated with a label, where a label is an element of a finite set known as an alphabet. A weighted graph is a labeled graph whose labels (weights) are real-valued and assigned to edges.

A $k$-regular graph (also known as a regular graph) is a graph where all the vertices have the same degree $k$. A complete graph is a graph where all the vertices are pairwise adjacent. In other words, a complete graph is a graph where all possible edges are present. We remark that a complete graph on $n$ vertices is an $n$-regular graph.

A tree is an acyclic graph that is connected. A spanning subgraph of a graph $G = (V, E)$ is a subgraph $G' = (V, E')$ of $G$ (i.e., $G$ and $G'$ have the same set $V$ of vertices). A spanning tree of a graph $G = (V, E)$ is a spanning subgraph $G' = (V, E')$ of $G$, where $G'$ a tree.

In Figure 2.2, we illustrate tree basic yet distinct graphs.

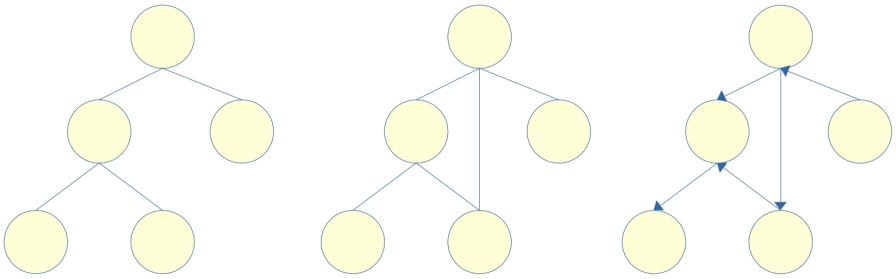

Figure 2.2: Illustration of three distinct graphs. The graph on the left is a graph that is a tree. The graph in the center is a graph with one additional edge (the additional edge makes the graph no longer a tree). The graph on the right is a directed graph (the arrows indicate the direction of edges).

### 2.2.1 Structural properties of graphs

In this subsection, we review some graph theory concepts that are related to structural properties of graphs, where by structural properties we mostly refer to the edge density or to regularities of the degree distribution. In particular, we review power-law degree sequences that are used in our contribution on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3). For the definitions in this subsection, we consult the textbook by van der Hofstad [Hof16].

We start by recalling definitions of the following graph theory concepts that are related to community structure: the connected component of a vertex, the largest connected component, and a giant component.

The connected component of a vertex $v \in V$ of a graph $G = (V, E)$ is a set $\mathcal{G}(v) \subseteq V$ of vertices, where $\mathcal{G}(v)$ includes a vertex $u \in V$ if there exists a path between $v$ and $u$ in $G$. The largest connected component $\mathcal{G}_{\max}$ of a graph $G$ is any connected component of a vertex $v \in V$ for which the cardinality of the connected component $\mathcal{G}(v)$ is maximal, i.e.,

$$\mathcal{G}_{\max} = \max_{v \in V} |\mathcal{G}(v)|.$$

A giant component of $G$ is the largest connected component $\mathcal{G}_{\max}$ of $G$ such that

$$\lim_{k \to \infty} \inf \frac{|\mathcal{G}_{\max}|}{k} > 0.$$

We proceed by discussing community structure.

As indicated by Leskovec et al. [Les+08], the definition of a community structure tends to vary from context to context. This is primarily because it is difficult to define a community in a network, where a network is a graph which has a stronger relation to real-world applications, and where the vertices of such graph are referred to as nodes or agents, and the edges are referred to as links. However, as suggested by the aforementioned authors, community detection is an actively studied area.

In this dissertation, we relate the community structure of a network to its structural properties, where by structural properties we either consider the edge density of two pairwise disjoint subjects of vertices, a clustering coefficient (whose definition tends to vary also), or regularities of the degree distribution. For example, when aiming for identifying or establishing a community structure in a network, the set of vertices of a graph could be partitioned in such a way that every partition class would contain the vertices of the same degree.

We recall the following definition of the clustering coefficient $c_G$ of a connected, undirected graph $G = (V, E)$:

$$c_G = \frac{1}{|V|} \sum_{v \in V} \frac{2|M(v)|}{|N(v)|(|N(v)| - 1)},$$

where, for each $v \in V$, the cardinality of the neighborhood of $v$ is $|N(v)| \geq 2$, and the set $M(v) \subseteq E$ includes all edges whose elements are in the neighborhood $N(v)$. As indicated by Watts and Strogatz [WS98], the value of the clustering coefficient $c_G$ indicates the cliquishness of a typical neighborhood in the graph $G$, where a clique of a graph $G$ is a complete subgraph $G'$ of at least three vertices such that $G'$ is not contained in a larger complete subgraph of $G$.

We proceed by discussing a small-world network and a scale-free network.

A small-world network is a network whose largest connected component contains a significant proportion of the vertices. The name of a small-world network is related to the small-world experiment [Mil67] which was a pioneering experiment on community structure in large real-world networks where nodes are persons, and the presence of a link between nodes indicate that the corresponding persons know each other. The experiment showed that the average shortest path in a network over a set of residents in the United States was unexpectedly short as the length of such path was estimated to be approximately 6. Furthermore, we highlight that, as indicated by Watts and Strogatz [WS98], the average shortest path in a small-world network is, typically, the logarithm of the number of nodes.

A scale-free network is a network whose degree distribution follows a power-law distribution, i.e., the relative frequency of a degree $d$ in the degree sequence is proportional to $d$ raised to a negative power. In other words, a power-law distribution is a probability distribution where the probability of observing a scalar value $x$ is proportional to $x$ raised to a negative power, i.e.,

$$p(x \,|\, a, \gamma) \propto a x^{-\gamma},$$

where $p$ denotes a probability density function, $a > 0$ and $\gamma > 0$. We recommend the article by Barabási and Albert [BA99] for a more extensive discussion on scale-free networks and power-law distributions.

Referring to the article by Boccaletti et al. [Boc+06], we recall and discuss a complex network and a chaotic network.

A complex network is a network that is large, non-regular, and dynamically evolves in time. Typically, a complex network models a real-world network that is large enough so that its properties are difficult to assess by taking its individual nodes. This way, the properties of a complex network are sometimes identified probabilistically using random graph models (discussed in Subsection 2.2.2). We remark that complex networks tend to have the characteristics of the aforementioned small-world network and the scale-free network.

The dynamic evolution mentioned in the definition of a complex network has an equilibrium state, where an equilibrium state is a state towards which the complex network evolves and then stops changing once it is reached. Otherwise, if an evolving network has no equilibrium state, we refer to it as a chaotic network. An example of a chaotic network is a sequence of graphs that repeats in cycles.

### 2.2.2 Random graph models

In this subsection, we review some common random graph models. In particular, we review the Erdős–Rényi model and the configuration model that are used, respectively, in our example and contribution on distributed averaging on random graphs (discussed, respectively, in Section 3.2 and Section 3.3).

We start by recalling definitions of a random graph model and a random graph.

A random graph model is a probability distribution over a family of random graphs, where a random graph is defined as a graph with fixed vertices and the presence of edges being modeled as a random variable.

A positive aspect of random graph models is that they allow for analysis of the properties of larger graphs (e.g., complex networks), primarily due to the applicability of techniques encountered in probability theory (as opposed to relying on the use of techniques encountered in combinatorics, which is another common practice in graph theory). For a review of random graph theory, we recommend and mostly follow the textbook by van der Hofstad [Hof16]. We remark that the textbook focuses on applications of random graph models for larger real-world networks.

We proceed by recalling definitions of the following random graph models: the Erdős–Rényi model, the stochastic block model, and the configuration model.

The Erdős–Rényi (ER) model [ER59] is a random graph model that is defined below.

**Definition 2.** *The ER model is a probability distribution over graphs, which is parametrized by the number $n$ of vertices and the probability $p$ of the edge assignment between any of its two vertices.*

We denote the probability mass function of the ER model by $p_G(G \mid n, p)$.

A graph that is an observation of an ER random graph has an $n \times n$ adjacency matrix with 0's in its main diagonal and with outcomes of independent Bernoulli trials of probability $p$ in the remaining elements of the adjacency matrix. In an ER

graph, the degree $d_v$ of a vertex $v$ follows the Binomial distribution, i.e., for each $d \in \{0, \ldots, n-1\}$, we have

$$\Pr(d_v = d) = \binom{n-1}{d} p^d (1-p)^{(n-1)-d}. \tag{2.7}$$

For larger ER graphs, the probability of a vertex $v$ taking a degree $d \in \{0, \ldots, n-1\}$ can be modeled by a Poisson distribution as follows:

$$\lim_{n\to\infty} \Pr(d_v = d) = \frac{(np)^d}{d!} e^{-np}.$$

We remark that the ER model was introduced at the end of the 50's by Erdős and Rényi [ER59]. Initially, the ER model was used to probabilistically prove the existence of a graph that meets certain properties, where such application of a random graph model is known as the probabilistic method [Alo+00]. In Figure 2.3, we illustrate two possible observations of an ER random graph, where the probabilities of the two observations differ.

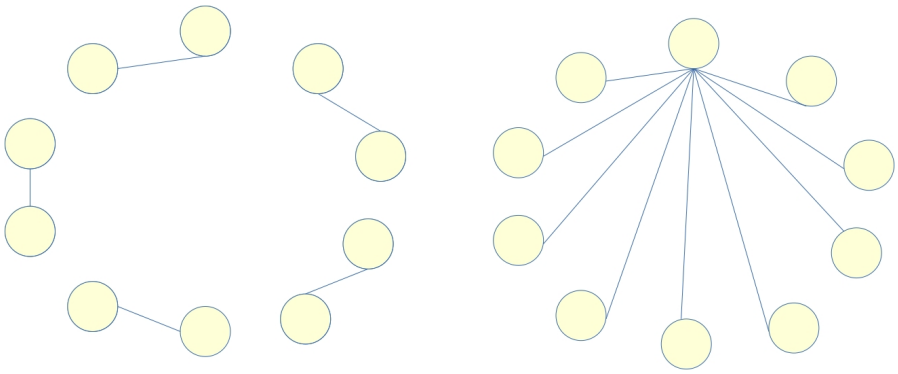

Figure 2.3: Illustration of two observations of the ER random graph, when $n = 10$ and $p = \frac{1}{10}$. For such ER random graph, as indicated by Equation 2.7, the probability of observing a vertex whose degree is 9 equals $\left(\frac{1}{10}\right)^9$, whereas the probability of observing a vertex whose degree is 1 equals $\frac{1}{10}\left(\frac{9}{10}\right)^8$. This suggests that an observation of a graph where all vertices have degree 1 is more likely.

The ER model has a phase transition in the presence of a giant component. We elaborate on this based on the analysis by Bollobás [Bol84]. Let $\epsilon \geq 0$. If $np \geq (1+\epsilon)$, then, with high probability, an ER random graph has a giant component of order at least $n^{2/3}$, and the order of any other connected component is $\mathcal{O}(\log n)$ (big O notation $\mathcal{O}$ is discussed in Subsection 2.3.1). Otherwise, if $np < (1+\epsilon)$, then, with high probability, there is no such giant component though the order of any connected component remains $\mathcal{O}(\log n)$.

The stochastic block model [HLL83] is a random graph model that is defined below.

**Definition 3.** *The stochastic block model is a probability distribution over graphs, which is parametrized by the number $n$ of vertices, the number $2 \leq k \leq n$ of communities (blocks), the expected fractions $\mathbf{n} \in [0,1]^k$ of vertices in a community, and the probabilities $\mathbf{P} \in [0,1]^{k\times k}$ for edge assignments between every two communities.*

We denote the probability mass function of the stochastic block model by $p_G(G \mid n, k, \mathbf{n}, \mathbf{P})$. For the previous definition, we consulted the textbook by van der Hofstad [Hof22].

The stochastic block model generalizes the ER model in such way that, instead of having a probability for edge assignments between any two vertices, we have probabilities for edge assignments between any two communities of vertices.

The configuration model [New03] is a random graph model that is defined as follows:

**Definition 4.** *The configuration model is a probability distribution over graphs, which is parametrized by a degree sequence* $\mathbf{d}$*, so that, for each* $i \in [n-1]$ *and* $j \in \{i+1, \ldots, n\}$*, we have*

$$\Pr\left(\{v_i, v_j\} \in E\right) = \frac{d_i d_j}{\left(\sum_{i' \in [n]} d_{i'}\right) - 1}.$$

We denote the probability mass function of the configuration model by $p_G(G \mid \mathbf{d})$. For the previous definition, we consulted the textbook by Newman [New10].

### 2.2.3 Basics of graph generation

In this subsection, we review some basics of graph generation. In particular, we review a procedure for generating graphs with power-law degree sequences, where the procedure is used in our contribution on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3).

Firstly, we review a general method for generation of mathematical objects known as inverse transform sampling. Then, we review some methods that are commonly encountered in graph generation.

Regarding graph generation, we mostly consider the case where the vertices of a graph are fixed and the presence of edges is determined by a random procedure, thus a generated graph can be interpreted as an observation of a random graph. We highlight that graph generation can have additional constraints, e.g., a generated graph must be connected. We remark that the additional constraints can sometimes be met fully and sometimes only partially but at the best effort.

In some contexts, other mathematical objects have to be generated first before generating a graph. For example, when generating a graph from the configuration model, we might need to generate a degree sequence first because the configuration model is parametrized by a degree sequence.

We proceed by discussing inverse transform sampling. For a review of inverse transform sampling, we recommend the textbook by Devroye [Dev86].

Inverse transform sampling (also known as the inversion method) is a universal strategy for generating mathematical objects (e.g., graphs or scalars) from a probability distribution, where such strategy requires a value $u$ that is independently observed from the uniform distribution, and which also requires the cumulative distribution function $\phi$ (for which we can compute the inverse) of the probability distribution. In particular, the generation of a value $x$ from the probability

distribution of interest by inverse transform sampling is the following evaluation:

$$x = \phi^{-1}(u),$$

where, for each generated value, we use an independent value $u$.

For example, if one aims for generating a power-law degree sequence $\mathbf{d}$, a common choice for this is the Pareto distribution which is a probability distribution with the following probability density function:

$$p(x \mid \alpha, \gamma) = \gamma \alpha^{\gamma} x^{-(\gamma+1)},$$

where $\alpha > 0$ is known as a scale parameter and $\gamma > 0$ is known as a shape parameter. For the definition of the Pareto distribution, we consulted the dictionary by Everitt and Skrondal [ES10]. For a more extensive description of the Pareto distribution, we recommend the article by Newman [New05].

For generating a degree sequence $\mathbf{d}$ of length $n$, we repeat inverse transform sampling $n$ times, thus for each $i \in [n]$, we compute the following independent value:

$$d'_i = \phi^{-1}_{\text{par}}(u_i),$$

where

$$\phi^{-1}_{\text{par}}(u_i) = \frac{\alpha}{u_i^{1/(\gamma-1)}}$$

is the inverse of the cumulative distribution function of the Pareto distribution and $u_i$ is the $i$-th observation of the uniform distribution. Since the degree is a natural number and not a real number, we would also perform rounding: for each $i \in [n]$, we have $d_i = \lceil d'_i \rceil$. This way, by collecting every degree $d_i$, we arrive to the intended generation of a power-law degree sequence $\mathbf{d}$. We remark that the generation of a graph that follows the configuration model is discussed in Chung and Lu [CL02].

We proceed by reviewing preferential attachment process which is a generation procedure carried iteratively along a number of parties (e.g., the parties could be the vertices of a graph), where the parties that have more resources (e.g., a higher degree of a vertex can be interpreted as a more numerous resource) are more likely to have the next resource allocated to them. Graphs that are generated by a preferential attachment process tend to have power-law degree sequences. We remark that preferential attachment processes are discussed more extensively in Barabási and Albert [BA99].

Finally, we review the Chinese restaurant process [ZC15] which is a particular preferential attachment process where every party can hold a limited number of resources. The Chinese restaurant process is commonly illustrated by an analogy where in a restaurant each table has a limited number of seats and a new customer can choose any table with an open seat, with a stronger preference for the tables with more occupied seats. To relate the analogy to graph generation, new edges are more likely to be assigned to vertices of higher degree. We remark that the Chinese restaurant process can be applied for generating graphs with an unfixed number of vertices. That is, in such case we allow for an event (with a certain probability) of the appearance of a new vertex as opposed to only allowing for events of appearances of edges. To relate to the analogy, the appearance of a new vertex corresponds to the opening of a new table.

## 2.3 Distributed systems

In this section, we review distributed systems. In particular, we review some preliminaries (e.g., randomized algorithms) for differential privacy (discussed in Subsection 2.5.1). Also, we review some preliminaries (e.g., overlay networks and gossip algorithms) for distributed averaging, which are discussed in our examples and contribution on the bias in distributed averaging when the graph of agents is non-regular (discussed in Chapter 3). Furthermore, we elaborate on a central curator which is present in our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4).

At the beginning of this section, we review some basic concepts related to distributed systems. In Subsection 2.3.1, we review some basic concepts related to algorithms. In Subsection 2.3.2, we discuss execution of distributed algorithms on graphs.

For a review of distributed systems, we recommend and mostly follow the textbook by van Steen and Tanenbaum [ST17].

We start reviewing the following basic concepts of distributed systems: a distributed system, a communication network, a communication protocol, and a central curator.

According to van Steen and Tanenbaum [ST17], definitions of a distributed system tend to vary from context to context. In this dissertation, we interpret a distributed system as a collection of autonomous computing elements that appears to its users as a single coherent system. We remark that the aforementioned definition does not take into account Byzantine failures which is a particular failure of distributed systems, where the cause of a failure of a single element of a distributed system can not be identified by the remaining elements of the distributed system.

More precisely, we interpret a distributed system as a communication network with a number of interconnected agents (e.g., machines, processors, processes) that store, process, and dispatch information according to a communication protocol, where a communication network refers to a network of agents which exchange messages, and where a communication protocol refers to a list of rules that are intended to be followed by the agents. In real-world distributed systems, the agents can be geographically distant, so that the communication itself is established by a far-reaching channel such as the internet.

A distributed system might have a central curator (also known as a central coordinator or simply a coordinator) which is an agent that has an assigned role for granting permissions to resources requested by other agents, where the permissions include the access to storing, processing, and forwarding of resources. Typically, a central curator is linked to every other agent of the communication network. We remark that Korach et al. [KKM90] provides an algorithm for choosing a central curator in a communication network (such procedure is known as the coordinator election problem or leader election).

In Figure 2.4, we illustrate an example of the graph of a distributed system without a central curator and an example of the graph of a distributed system

which strongly relies on a central curator.

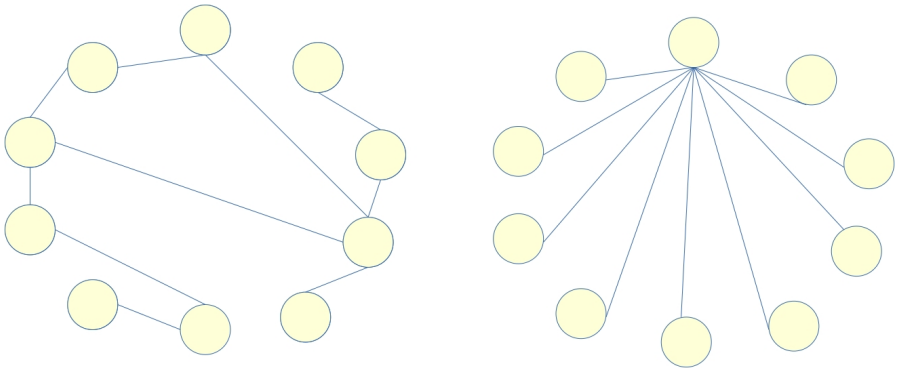

Figure 2.4: Illustration of a distributed system without a central curator (left) and a distributed system with a central curator (right). We remark that the central curator is the node to which every other node is connected. In the distributed system on the left, the message of a sender can traverse several intermediate agents until the addressee receives the message. In the distributed system on the right, the message traverses only the central curator (the longest path that a message can travel is of length 2).

We proceed by discussing an overlay network.

An overlay network of a communication network is a communication network that is built "on top" or "in parallel" to the formerly mentioned communication network. We remark that the graph of an overlay network is usually connected. Also, we remark that overlay networks are useful for addressing robustness issues. For example, when a communication network is susceptible to inactive agents, an overlay network could be built in such a way that each agent would have more or at least several neighbors. This way, if one neighbor becomes inactive, the information can still flow along the remaining active neighbors. For a more extensive discussion on overlay networks, we recommend the article by Sitaraman et al. [Sit+14]

We proceed by discussing some particular modes of information flow in distributed systems. This mostly concerns approaches of how and when the agents exchange, share, or access messages.

Firstly, we highlight that we focus on distributed systems where agents collaboratively solve common tasks, i.e., the agents share resources with the aim of reaching common objectives. For example, contribution and revision of Wikipedia articles can be interpreted as an example of a distributed system characterized by collaboration. In this dissertation, we in particular focus on the task where each agent intends to obtain the average of values distributed over the agents of a communication network.

Secondly, we highlight that we also focus on distributed systems with the following characteristics of self-management: minimized reliance on a central curator, handshake-free interaction, asynchronicity. This way, we proceed by discussing the following concepts related to information flow in a communication network: a handshake, synchronicity, and asynchronicity.

When speaking of handshakes, we refer to tuples of two messages, i.e., a request

(of a task) together with a reply, where the request is sent from an agent $i$ to an agent $j$, and the reply (sent from the agent $j$ to the agent $i$) confirms that the task was carried successfully, which is similar to handshakes discussed in Denning and Sacco [DS81]. For example, handshakes can be used to improve certainty that a request was received, since the reply to the request can be a message indicating that the request was received. However, we remark that the two generals' problem [AEH75] is a common example of distributed systems, which illustrates that a communication protocol which relies on handshakes is insufficient for having complete certainty that a message from one agent was reached by another.

Synchronicity is a mode of communication where messages among participants (e.g, agents of a distributed system) are sent at the same (or near-real) time. Asynchronicity is a mode of communication where messages among participants are sent at different times. For the previous definitions, we consulted Mick and Middlebrook [MM15].

We remark that synchronous communication and asynchronous communication can be modeled by the ticks of a clock, where the ticks indicate the instances of time at which the communication (or processing) is performed. We can model synchronous communication by a global clock, where every agent communicates at the same instances of time as the other agents. Regarding asynchronous communication, we can model it by having an independent local clock for each agent, thus the agents can communicate at distinct instances of time.

Asynchronous communication and synchronous communication may not be fully realistic or practical. In asynchronous communication, an agent communicates to its neighboring agents one at a time, and thus it may be unpractical to wait for the handshakes from a neighbor that is either inactive or at that moment is executing a significantly costly process. On the other hand, in synchronous communication, it may happen that the interactions finished out of phase (some agents had no more messages to send but not the others), and hence one can not trust that all computations are complete.

### 2.3.1 Algorithms

In this subsection, we review some basic concepts related to algorithms. In particular, we review some preliminaries of distributed algorithms that are used in our example where a distributed algorithm is executed on a graph (discussed in Subsection 2.3.2). Also, we review randomized algorithms and gossip algorithms, which are used in our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4).

For the definitions in this subsection, we consider the lecture notes by Hodkinson and Cunninghamas [HC91]. For a supplementary reference to definitions related to algorithms, we recommend the textbook by Cormen et al. [Cor+09]

We start by recalling a definition of a Turing machine.

A Turing machine is a 6-tuple $M = (S, \Sigma, \Sigma_0, s_0, F, f)$, where

- $S$ is a finite non-empty set, where an element of $S$ is known as a state.

- $\Sigma$ is a finite set of at least two elements, where $\Sigma$ is known as the alphabet of $M$. It is required that a particular element $b \in \Sigma$, where $b$ is known as the blank symbol.
- $\Sigma_0 \subseteq \Sigma \setminus \{b\}$ is a non-empty set, where $\Sigma_0$ is known as the input alphabet of $M$.
- $s_0 \in S$ is known as the initial state of $M$.
- $F \subset S$ is known as the set of final states of $M$.
- $f : (S \setminus F) \times \Sigma \to S \times \Sigma \times \{-1, 0, 1\}$ is a partial function, where $f$ is known as the instruction table of $M$.

We remark that a Turing machine can be interpreted as a 1-way infinite tape, a read/write head, and a finite set of states. A Turing machine can *get* its current state, *read* the current state, and then *write*, *move*, and *change* state according to its instruction table. The blank symbol is the only symbol in the alphabet that can occur on the tape infinitely many times. This way, we highlight that an algorithm is what a Turing machine implements.

In this dissertation, we sometimes simplify the interpretation of an algorithm, and thus consider it as a sequence of instructions aimed to perform a particular task. Sometimes, we express an algorithm as a function whose input is accessible to all instructions of the algorithm and whose output is the result of the last instruction of the algorithm.

We proceed by recalling definitions of the computation cost and the asymptotic time complexity.

The computational cost (also known as the running time) of an algorithm is the amount of time it takes to execute the algorithm. For example, let us consider a random walk on a graph, which, referring to Lovász [Lov93], is a finite Markov chain (discussed in Subsection 2.4.3). In other words, a random walk on a graph is a stochastic process that describes a walk on a graph. If we interpreted a random walk as an algorithm, the computational cost of a random walk can be considered as the length of the walk that was taken until the random walk terminated.

When the computational cost of an algorithm is difficult to identify precisely, sometimes it can be helpful to consider the asymptotic time complexity of an algorithm, where the asymptotic time complexity can be expressed in big O notation. Let functions $f, g : \mathbb{R} \to \mathbb{R}$. Big O notation $\mathcal{O}(\cdot)$ is expressed as follows:

$$f(x) = \mathcal{O}\big(g(x)\big) \quad (\text{as } x \to \infty),$$

which means that there exists $k > 0$ and $x_0 \in \mathbb{R}$ so that, for each $x > x_0$, we have

$$f(x) \leq kg(x).$$

In other words, big O notation is an asymptotic upper bound at the limit of arguments of a function going to infinity. In graph problems, the asymptotic time complexity of an algorithm is commonly expressed by an upper bound when the number of vertices of a graph or the number of edges of a graph goes to infinity.

We proceed by recalling definitions of a recursive algorithm, a randomized algorithm, a distributed algorithm, and a gossip algorithm.

A recursive algorithm is an algorithm which has at least one instruction that reuses the algorithm itself (such reuse is known as a nesting). This way, the recursion depth is defined as the maximal number of nestings within nestings. The recursion depth of an algorithm can be interpreted as a counter that decreases by 1 once the furthest nesting terminates. When the counter reaches 0, there are no more nestings to execute.

A randomized algorithm is an algorithm which uses random variables. For the previous definition, we consulted Motwani and Raghavan [MR96].

A distributed algorithm is an algorithm which is executed in a distributed system. For example, a random walk on a graph is a distributed and randomized algorithm that models the traversal of a message in a communication network. For an elaborated discussion of distributed algorithms, we recommend the textbook by Tel [Tel00].

We discuss an example of a distributed, recursive, randomized algorithm that collects a sample of vertices of a graph $G$. Let $r \in \mathbb{N}_1$ be the recursion depth. For each $v \in V$, let $U_v \subseteq N(v)$ be a uniformly-random choice of a subset of vertices from the neighborhood $N(v)$. (We remark that $U_v$ contains scalars and not random variables.) Let a distributed, recursive, randomized algorithm $\mathcal{A}$ be defined as follows:

$$\mathcal{A}(v, r) = \begin{cases} \{v\} & \text{when } r = 0, \\ \cup_{u \in U_v} \mathcal{A}(u, r-1) \cup \{v\} & \text{when } r > 0, \end{cases}$$

where $\cup$ denotes the union of sets and $\cup_{u \in U_v} \mathcal{A}(u, r-1)$ denotes the union of $\mathcal{A}(u, r-1)$ over each $u \in U_v$. In the first step, the algorithm $\mathcal{A}$ appends the initial vertex $v$ to the output set. Then, $\mathcal{A}$ propagates along the selected neighbors, adds them to the output set, and calls itself again with a decreased counter. This happens for $r$ recursive iterations in total. If $r$ is large enough, the output set contains all vertices of $G$ if $G$ is connected. This is because the algorithm propagates along every vertex of $G$.

As suggested by Jelasity [Jel11], it is difficult to capture what exactly is a gossip algorithm. In this dissertation, we consider that a gossip algorithm is a distributed algorithm, where each agent of the communication network can disseminate or aggregate information along its neighboring agents. We provide a reference to Demers et al. [Dem+87] which is known as one of the earliest formalizations of a gossip algorithm.

### 2.3.2 Basics of distributed algorithms on graphs

In this subsection, we discuss execution of distributed algorithms on graphs. In particular, we briefly review distributed averaging and discuss some assumptions, so that information at one agent of the communication network could at some point reach any other agent of the communication network, which is used in our example and contribution on the bias in distributed averaging when the graph of agents is non-regular (discussed in Chapter 3) Afterwards, we discuss an example where

a random walk on the graph of a communication network provides a uniformly-random sample of a vertex of that graph. We remark that the aforementioned example illustrates the basis of the problem studied in Chapter 3.

Distributed averaging is a task where every agent of a communication network may communicate only with its neighboring agents, while aiming to compute the average of individual values attributed to all agents in the communication network. One of more common methods for solving such distributed averaging problems is an application of gossip algorithms [SHS19]. We remark that computing such averages is a basis for numerous applications that involve decision making or personalized modeling [AX18]. We remark that decision making in distributed systems without a central curator is discussed in Tsitsiklis [Tsi84].

We review two common assumptions on the graph of a communication network for executing distributed algorithms, so that information at one agent of the communication network could at some point reach any other agent of the communication network.

The first assumption is that the graph $G$ of a communication network is connected. The second assumption is that $G$ has a self-loop or a cycle of odd length. For the source of the two assumptions, we refer to DeGroot learning [DeG74], where DeGroot learning refers to a particular model of distributed averaging.

Referring to the article by Lovász [Lov93], we discuss an example where a random walk samples a vertex of a graph $G = (V, E)$. In particular, we show that a naive application of a random walk results in a biased sample, when $G$ is non-regular (when the vertices have unequal degrees).

Let $v_{\text{beg}}$ denote the vertex on which the random walk begins. Let $v_{\text{end}}$ denote the vertex on which the random walk ends ($v_{\text{end}}$ is the sampled vertex). Let $s \in \mathbb{N}$ be the number of steps the random walk takes ($s$ is a counter). Let $n = |V|$ denote the number of vertices. Let $\mathbf{T}$ be an $n \times n$ matrix where, for each $i, j \in [n]$, we have

$$t_{i,j} = \begin{cases} 1/\deg(v_i) & \text{when } v_j \in N(v_i), \\ 0 & \text{otherwise.} \end{cases}$$

In DeGroot learning [DeG74], $\mathbf{T}$ is known as a transition matrix, where, for each $i, j \in [n]$, the element $t_{i,j}$ of $\mathbf{T}$ is the probability of the transition from the vertex $v_i$ to the vertex $v_j$, where $v_i, v_j \in V$. When each row of a transition matrix sums to 1, the transition matrix is a row stochastic matrix. When each row and each column of a transition matrix sums to 1, the transition matrix is a doubly stochastic matrix. We remark that the transition matrix $\mathbf{T}$ is a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^n$. For the definitions of stochastic matrices, we consulted the textbook by Horn and Johnson [HJ12].

Let $p(v \mid s, v_{\text{beg}})$ denote the probability of sampling $v \in V$ at the step $s$, when the random walk begins on the vertex $v_{\text{beg}}$. Since the initial vertex $v_{\text{beg}}$ is chosen uniformly, let $p(v_{\text{beg}})$ be defined as a uniform probability mass function over the vertices in $V$. We define $p(v \mid s, v_{\text{beg}})$ by applying the transition matrix $\mathbf{T}$ iteratively as follows:

$$\begin{aligned} p(v \mid s, v_{\text{beg}}) &= \mathbf{T} p(v \mid s-1, v_{\text{beg}}) \\ &= \mathbf{T}^s p(v_{\text{beg}}). \end{aligned}$$

This way, $p(v \mid s, v_{\text{beg}})$ models a random walk that samples a vertex $v$ of a graph $G$.

In Levin et al. [LPW17], it is claimed that $s = \mathcal{O}(\log n)$ is enough so that $p(v \mid s, v_{\text{beg}})$ is a stationary distribution over the vertices in $V$, when $G$ is connected and has a self-loop or a cycle of odd length. For random walks, such stationary condition is more commonly known as the mixing time. When the mixing time is reached, we have

$$\Pr(v_{\text{end}} \mid s, v_{\text{beg}}) = \frac{\deg(v_{\text{end}})}{2|E|},$$

i.e., the probability of sampling $v_{\text{end}}$ is non-uniform when $G$ is non-regular. In other words, the sampling is biased when $G$ is non-regular.

We recall how the aforementioned application of the random walk can be improved using the Metropolis–Hastings algorithm (discussed in Subsection 2.4.3) so that any vertex of $G$ is sampled uniformly. The main idea behind such Metropolis–Hastings step is that, when the random walk proposes, for example, the edge $(v_1, v_2)$ which is the transition of the random walk from a vertex $v_1$ to a vertex $v_2$, then we perform a Bernoulli trial for deciding if the edge is indeed taken or another edge proposal is drawn (and an additional check of a Bernoulli trial is made). The probability of succeeding such Bernoulli trial is defined as follows:

$$\min\left(1, \frac{\Pr(v_1 \mid s, v_{\text{beg}})}{\Pr(v_2 \mid s, v_{\text{beg}})}\right) = \min\left(1, \frac{\deg(v_1)}{\deg(v_2)}\right).$$

As discussed later, upon the Metropolis–Hastings step and upon reaching the mixing time, the random walk provides a uniform sample of a vertex when $G$ is non-regular. In other words, the Metropolis–Hastings step corrects the bias when $G$ is non-regular.

## 2.4 Machine learning

In this section, we review machine learning. In particular, we review linear regression models used in our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4). Also, we review the Metropolis–Hastings algorithm which we use in our examples on the bias of sampling a vertex of a graph using a random walk and in distributed averaging when the graph of agents is non-regular (discussed, respectively, in Subsection 2.3.2 and Subsection 3.1.2).

At the beginning of this section, we review definitions of basic concepts of machine learning. In Subsection 2.4.1, we discuss supervised learning. In Subsection 2.4.2, we discuss clustering. In Subsection 2.4.3, we discuss statistical inference.

For a review of machine learning, we recommend and mostly follow the textbook by Mohri et al. [MRT18] For a supplementary reference to definitions of machine learning, we recommend the textbook by Mitchell [Mit97].

Mitchell [Mit97] defines machine learning as the study of algorithms that allows for computer programs to automatically improve through experience. This way, a learner (which is a computer program) is said to learn from experience with

respect to some class of tasks and performance measure, if its performance at the class of tasks, as measured by the performance measure, improves with the experience. For example, this involves the case where the learner uses previously acquired information and current information for an accurate modeling of certain phenomena.

We start by discussing the following basic concepts of machine learning: an attribute, a feature, an example, and a label.

We interpret an attribute as a base characteristic of an entity. However, we remark that it is difficult to capture the definition of an attribute exactly, as it tends to vary from context to context. We interpret a feature as a value describing a characteristic of an entity, where such description is useful for tasks related to prediction, inference, and evaluation, and where a feature is computed from other attributes or features of an entity. We remark that we consider restrained definitions of attributes and features in order to clarify the motivation of our contribution on the tailored noise mechanism (discussed in Chapter 4).

An example (also known as an instance) is a representation of an entity, it may be a feature or a feature vector. A label (also known as a target value) is a value assigned to an example, which indicates a correspondence to a particular task. A pair of an example and a label is known as a labeled example. For instance, if an example contains characteristics of an email, a label could be an indicator of whether the email is spam or not. We denote a set of (all possible) examples by $\mathcal{X}$, and we denote a set of (all possible) labels by $\mathcal{Y}$.

We proceed by recalling definitions of a concept and a hypothesis.

A concept is a function $c : \mathcal{X} \to \mathcal{Y}$ interpreted as an unknown function that models a phenomenon. A concept class is a set of concepts of interest; we denote a concept class by $\mathcal{C}$. A hypothesis is a function $h : \mathcal{X} \to \mathcal{Y}$ interpreted as a model of a concept. A hypothesis space is a set of hypotheses; we denote a hypothesis space by $\mathcal{H}$. For example, the set $\mathcal{H}$ of functions $f(x) = \theta_0 + \theta_1 x$ over parameters $\theta_0, \theta_1 \in \mathbb{R}$ is a hypothesis space of linear functions.

Referring to the lecture notes by Rosernberg [Ros16], we discuss the evaluation of how closely a hypothesis matches a concept. This way, we recall definitions of a loss function, the hypothesis risk, the empirical risk, and the empirical risk minimizer.

A loss function is a function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ that weights the cost of a choice of a hypothesis to model a given concept, where $\mathbb{R}^+$ is the space of non-negative real numbers.

The hypothesis risk is defined as follows:

$$r(h, c) = \int_{\mathcal{X}} \mathcal{L}\big(h(x), c(x)\big) dx,$$

where $h$ is a hypothesis, $c$ is a concept, $\mathcal{L}$ is a loss function, and $\mathcal{X}$ is a set of (all possible) examples. We remark that the hypothesis risk can not be computed because the concept $c$ is unknown.

The empirical risk is defined as follows:

$$\hat{r}\big(h, \{(x_i, y_i)\}_{i=1}^n\big) = \frac{1}{n} \sum_{i \in [n]} \mathcal{L}\big(h(x_i), y_i\big),$$

where the set $\{(x_i, y_i)\}_{i=1}^n$ of labeled examples is a sample from a statistical population over $\mathcal{X} \times \mathcal{Y}$, and where $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}$ denote, respectively, a set of (all possible) examples and a set of (all possible) labels.

An empirical risk minimizer is a hypothesis defined as follows:

$$\hat{h} \in \arg\min_{h \in \mathcal{H}} \hat{r}\big(h, \{(x_i, y_i)\}_{i=1}^n\big),$$

where $\mathcal{H}$ is a hypothesis space.

In Table 2.4, we provide a summary of the introduced notation of basic concepts of machine learning.

Table 2.4: Introduced notation of basic concepts of machine learning

| Notation | Meaning |
|---|---|
| $\mathcal{X}$ | Set of (all possible) examples |
| $\mathcal{Y}$ | Set of (all possible) labels |
| $\mathcal{L}$ | Loss function |
| $\mathcal{C}$ | Concept space |
| $\mathcal{H}$ | Hypothesis space |
| $h$ | Hypothesis |
| $\hat{h}$ | Empirical risk minimizer |
| $r$ | Hypothesis risk |
| $\hat{r}$ | Empirical risk |

Referring to the lecture notes by Rosernberg [Ros16], we discuss the hypothesis space error and the sampling error.

The hypothesis space error is the error due to selecting a particular hypothesis space and is defined as follows:

$$r(h_{\mathcal{H}}, c),$$

where a risk minimizer $h_{\mathcal{H}}$ is defined as

$$h_{\mathcal{H}} \in \arg\min_{h \in \mathcal{H}} r(h, c).$$

The hypothesis space error $r(h_{\mathcal{H}}, c)$ can not be computed because the concept $c$ is unknown.

The observation error (also known as the sampling error) is the error due to limited number of observations and is defined as follows:

$$\hat{r}\big(\hat{h}, \{(x_i, y_i)\}_{i=1}^n\big) - r(h_{\mathcal{H}}, c).$$

The observation error can not be computed because, as discussed above, we can not compute the hypothesis space error $r(h_{\mathcal{H}}, c)$. However, we remark that the observation error can be approximated by upper-bounding it by the empirical risk $\hat{r}\big(\hat{h}, \{(x_i, y_i)\}_{i=1}^n\big)$.

### 2.4.1 Supervised learning

In this subsection, we discuss supervised learning. In particular, we review linear regression which is used in our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4).

We start by briefly reviewing supervised learning. Then, we discuss the fitting of statistical models. Finally, we discuss classification and regression, which are two common tasks related to supervised learning.

Supervised learning is a machine learning scenario where the learner has a set of labeled examples and a set of unlabeled examples, and where the learner intends to use the labeled examples to predict the labels for the unlabeled examples.

Referring to the textbook by Mitchell [Mit97], we discuss the fitting of statistical models. In particular, we discuss fitting, overfitting, and underfitting.

Let us consider the case where the learner uses labeled examples to estimate parameters of an underlying statistical model with the intention to predict the labels for unlabeled examples with sufficient accuracy. An estimation procedure that makes use of labeled examples to estimate model parameters is known as the fitting of a statistical model.

Regarding the accuracy of a fitted statistical model, it is influenced by the number of labeled examples used for fitting, the choice of a hypothesis space, and the choice of a fitting method. We remark that having more labeled examples leads to closer estimates of the parameters of an underlying statistical model because the sampling error decreases when the size of a sample increases. Also, we remark that a chosen hypothesis space intends to contain statistical models (hypotheses) that are sufficiently close to the underlying statical model (the concept).

Given a hypothesis space $\mathcal{H}$, a hypothesis $h \in \mathcal{H}$ is said to overfit a sample if there exists some other hypothesis $h' \in \mathcal{H}$, such that $h$ has a smaller error than $h'$ over the sample, but $h'$ has a smaller error over the entire underlying statistical population. This way, overfitting is the case when the choice of a hypothesis $h$ poorly generalizes the underlying concept $c$. Otherwise, when a hypothesis $h$ has a smaller error over a sample than some other hypothesis $h'$ because $h$ is a function that lacks significant terms compared to the underlying concept $c$ (e.g., $h$ is a polynomial of a lower order than $c$), then the case of choosing such $h$ is known as underfitting.

We proceed by discussing the evaluation of fitted statistical models.

The fit of a statistical model can be evaluated by assessing the utility (e.g., predictive accuracy) of the chosen hypothesis by splitting the available set of labeled examples to two subsets known as a training set and a test set. To elaborate on that process, we remark that, firstly, a statistical model is fitted using the training set. Then, we can compute predicted labels of the test set. Finally, the fit can be evaluated by, for example, summing the absolute differences between the predicted labels of the test set and the respective true labels of the test test.

The evaluation of a fit is more reliable when performing it on several choices of training sets and test sets, this is known as cross-validation. A common cross-validation technique is $k$-fold cross-validation where the set of labeled examples is arbitrarily partitioned in $k$ subsets (folds) of approximately equal size and the

fitted statistical model is evaluated $k$ times by taking 1 of those folds as a test set and forming a training set from the remaining $k-1$ folds.

We proceed by making a brief distinction between classification and regression.

Classification is a supervised learning task where the labels are categorical (e.g., binary values), whereas regression is a supervised learning task where the labels are real-valued. We highlight that, in this dissertation, we focus on a particular form of linear regression where a target value is expressed as a linear function of features together with a noise term. More precisely, we consider multiple linear regression where a target value is expressed as a linear combination of features together with a noise term, where each feature is multiplied by a scalar parameter.

We proceed by discussing regression in more detail, for which we consult the textbook by Yan and Su [YS09]. In particular, we start by recalling a definition of a multiple linear regression model. Then, we discuss two classic fitting methods for fitting a linear regression model. Finally, we discuss an example of evaluating the fit of a linear regression model.

We recall a definition of a multiple linear regression model with $m+1$ regression parameters and a scalar target value. Let $n \in \mathbb{N}$ be the number of examples. Let $m \in \mathbb{N}$ be the number of features in an example (i.e., the size of a feature vector). For each $i \in [n]$, we have

$$y_i = \theta_0 + \theta_1 x_{i,1} + \ldots + \theta_m x_{i,m} + \xi_i^{\text{reg}}, \tag{2.8}$$

where $x_{i,1}, x_{i,2}, \ldots, x_{i,m}$ are features, $\theta_0, \theta_1, \ldots, \theta_m \in \mathbb{R}$ are regression parameters, $\xi_i^{\text{reg}}$ is regression noise which is an independent observation of $\mathcal{N}(0, \sigma^2)$, and $\sigma$ is the standard deviation of regression noise.

We recall ordinary least squares which is a classic fitting method of a multiple linear regression model. Let $\hat{\theta}$ be the vector of parameter estimates. Let $\mathbf{X} = \begin{bmatrix}\mathbf{1} & \mathbf{x}_1 & \ldots & \mathbf{x}_m\end{bmatrix} \in \mathbb{R}^{n \times (m+1)}$ be a matrix of features, where $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ are feature vectors. Let $\mathbf{y}$ be a vector of target values. In ordinary least squares, the vector $\hat{\theta}$ of parameter estimates can derived in the following way (the regression error terms are absent as they can not be observed):

$$\begin{aligned}
\mathbf{y} = \mathbf{X}\hat{\theta} &\iff \mathbf{X}^{\mathrm{T}}\mathbf{y} = \mathbf{X}^{\mathrm{T}}\mathbf{X}\hat{\theta} \\
&\iff \hat{\theta} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \\
&\iff \hat{\theta} = \left(\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{y},
\end{aligned} \tag{2.9}$$

$$\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X} = \begin{bmatrix}
1 & \frac{1}{n}\sum_{i\in[n]} x_{i,1} & \ldots & \frac{1}{n}\sum_{i\in[n]} x_{i,m} \\
\frac{1}{n}\sum_{i\in[n]} x_{i,1} & \frac{1}{n}\sum_{i\in[n]} x_{i,1}^2 & \ddots & \frac{1}{n}\sum_{i\in[n]} x_{i,1}x_{i,m} \\
\vdots & \ddots & \ddots & \vdots \\
\frac{1}{n}\sum_{i\in[n]} x_{i,m} & \frac{1}{n}\sum_{i\in[n]} x_{i,1}x_{i,m} & \ldots & \frac{1}{n}\sum_{i\in[n]} x_{i,m}^2
\end{bmatrix} \tag{2.10}$$

$$\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{y} = \begin{bmatrix}\frac{1}{n}\sum_{i\in[n]} y_i & \frac{1}{n}\sum_{i\in[n]} y_i x_{i,1} & \ldots & \frac{1}{n}\sum_{i\in[n]} y_i x_{i,m}\end{bmatrix}^{\mathrm{T}}. \tag{2.11}$$

Since the matrix inverse $\left(\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}$ can be numerically difficult to compute, we recall another classic fitting method known as regularized least squares (otherwise

known as ridge regression) where the term $\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is substituted with the term

$$\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_{m+1} = \begin{bmatrix} 1+\lambda & \dots & \frac{1}{n}\sum_{i\in[n]} x_{i,m} \\ \frac{1}{n}\sum_{i\in[n]} x_{i,1} & \dots & \frac{1}{n}\sum_{i\in[n]} x_{i,1}x_{i,m} \\ \vdots & \ddots & \vdots \\ \frac{1}{n}\sum_{i\in[n]} x_{i,m} & \dots & \frac{1}{n}\sum_{i\in[n]} x_{i,m}^2 + \lambda \end{bmatrix}, \tag{2.12}$$

and where $\lambda \in \mathbb{R}$ is known as the regularization parameter, and $\mathbf{I}_{m+1}$ is the $(m+1)\times(m+1)$ identity matrix.

We discuss an example of evaluating the fit of a linear regression model Let $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ be a set of labeled examples (pairs of features and target values). For each $i \in [n]$, we define the following linear regression model:

$$y_i = \theta_0 + \theta_1 x_i + \xi_i^{\mathrm{reg}},$$

where $\theta_0, \theta_1 \in \mathbb{R}$ are regression parameters, $\xi_i^{\mathrm{reg}}$ is regression noise which is an independent observation of $\mathcal{N}(0, \sigma^2)$, and $\sigma$ is the standard deviation of regression noise. Let $\{(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)\}$ and $\{(x_k, y_k), (x_{k+1}, y_{k+1}), \dots, (x_n, y_n)\}$ be, respectively, a training set and a test set, where $1 < k < n$. Let $\hat{\theta}_0$ and $\hat{\theta}_1$ be the parameter estimates computed from the training set by some fitting method (e.g., ordinary least squares). This way, for each $i \in \{k, \dots, n\}$, we compute a predicted target value $y_i^{\mathrm{pred}}$ as follows:

$$y_i^{\mathrm{pred}} = \hat{\theta}_0 + \hat{\theta}_1 x_i.$$

Finally, we can evaluate the fit of the learned model with the parameter estimates $\hat{\theta}_0$ and $\hat{\theta}_1$ by assessing its predictive accuracy. This way, for example, we can compute the following sum over the test set:

$$\sum_{i\in\{k,\dots,n\}} |y_i^{\mathrm{pred}} - y_i|.$$

However, we remind that evaluation of a fit is more reliable when applying the previously discussed cross validation.

### 2.4.2 Clustering

In this subsection, we discuss clustering. In particular, we review $k$-means clustering to which we refer in our future directions (discussed in Section 5.2).

For the definitions in this subsection, we consult the textbook by Shalev-Shwartz and Ben-David [SB14]

We start by briefly reviewing clustering which is part of a machine learning scenario known as unsupervised learning. We remark that in unsupervised learning, differently than in supervised learning, we have only unlabeled examples.

Clustering is a machine learning task where we have only unlabeled examples, where similar mathematical objects are grouped together in clusters (or partition classes) and dissimilar objects are grouped in separate clusters (or separate partition classes), and where distinct clusters can share the same instances (this leads to soft

clustering which contrasts hard clustering where distinct partition classes do not share the same instances).

The assignment of an object to a cluster (or a partition class) can be interpreted as the assignment of a label that indicates the index of the cluster. As a result, in clustering, the labels simply indicate indexing, as opposed to some meaningful classes (which is the case in the task of classification). This way, it is difficult to apply a general technique to evaluate the success of a clustering procedure. We remark that even though a clustering procedure can provide a set of clusters that is informative, there might be other sets of clusters that are informative as well.

Common clustering methods include linkage-based clustering, $k$-means clustering, and spectral clustering:

- Linkage-based clustering methods are iterative methods where clusters merge if their similarity is high enough.
- $k$-means clustering methods are iterative methods where instances are associated to one of $k$ clusters based on cost (e.g., distance) minimization.
- Spectral clustering methods are methods that group instances based on the eigenvalues of a similarity matrix of instances, where the $(i, j)$-th element of a similarity matrix is a similarity measurement between an instance $i$ and an instance $j$.

We elaborate on $k$-means clustering methods because we refer to them upon discussing our future directions (discussed in Section 5.2).

$k$-means clustering is an iterative clustering method which partitions instances (with coordinates) into $k$ partition classes based on center points (the initial coordinates of the center points of the $k$ partition classes can be chosen arbitrarily), and one iteration of $k$-means clustering takes the following two steps:

1. Each clustered instance is assigned the label of the center point that has the shortest distance to the instance (the distance is computed based on the coordinates and a chosen metric).
2. The locations of the center points are updated taking the average values of the coordinates of the instances that are assigned to the respective center points.

### 2.4.3 Statistical inference

In this subsection, we discuss statistical inference. In particular, we review the Metropolis–Hastings algorithm which we use in our examples on the bias of sampling a vertex of a graph using a random walk and in distributed averaging when the graph of agents is non-regular (discussed, respectively, in Subsection 2.3.2 and Subsection 3.1.2).

For the definitions in this subsection, we consult the textbook by Hastie et al. [HTF09]

We start by reviewing statistical inference, where statistical inference is part of artificial intelligence which is a field that is broader than machine learning.

Statistical inference (also known as inference) is a task where probabilistic information is inferred from evidence. Let $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be a sample of labeled examples (also known as evidence). Let $\mathcal{H}$ denote a hypothesis space. Common statistical inference methods include maximum likelihood inference and Bayesian inference:

- Maximum likelihood inference involves methods that aim for obtaining a hypothesis $h_{\mathrm{L}} = \arg\max_{h \in \mathcal{H}} \Pr(\mathcal{D} \mid h)$. In other words, such methods choose the hypothesis $h_{\mathrm{L}}$ for which the likelihood (defined below) of observing the evidence $\mathcal{D}$ is maximized.

- Bayesian inference involves methods that aim for obtaining a hypothesis $h_{\mathrm{P}} = \arg\max_{h \in \mathcal{H}} \Pr(h \mid \mathcal{D})$. In other words, such methods choose the hypothesis $h_{\mathrm{P}}$ that maximizes the posterior (defined below) upon the evidence $\mathcal{D}$.

We proceed by reviewing Bayesian inference in more detail.

Bayesian inference is a form of statistical inference characterized by an application of Bayes' theorem and by the treatment of the output and the parameters of statistical models as random variables. Let random variables $X$ and $\Theta$ be, respectively, the output of a statistical model and the parameter of the statistical model. In Bayesian inference, the relation between $X$ and $\Theta$ is established by Bayes' theorem as follows:

$$\begin{aligned} p_\Theta(\theta \mid x) &= \frac{p_{X,\Theta}(x, \theta)}{p_X(x)} \\ &= \frac{p_{X|\Theta}(x \mid \theta) p_\Theta(\theta)}{p_X(x)} \\ &= \frac{p_{X|\Theta}(x \mid \theta) p_\Theta(\theta)}{\int p_{X|\Theta}(x \mid \theta) p_\Theta(\theta) d\theta}, \end{aligned}$$

where $p_X$ is known as a prior probability distribution (or simply a prior), $p_{\Theta \mid X}$ is known as a posterior distribution (or simply a posterior), $p_{X|\Theta}(x \mid \theta)$ is known as a likelihood, and $\int p_{X|\Theta}(x \mid \theta) p_\Theta(\theta) d\theta$ is known as a marginal likelihood.

The marginal likelihood can be difficult to compute and thus may require to restate the Bayesian inference problem. This way, in Bayesian inference, it is common to estimate the posterior rather than obtain it exactly. For example, variational methods is a family of such methods, where a variation (a probability distribution) substitutes the likelihood in such a way that the integration in the marginal likelihood has a closed-form expression. Hence, the marginal likelihood appears feasible to compute in practical settings.

We proceed by discussing Markov chain Monte Carlo methods which is another method for estimating the posterior. Then, we review a definition of a particular Markov chain Monte Carlo method known as the Metropolis–Hastings algorithm.

A Markov chain Monte Carlo (MCMC) method is an algorithm that is characterized by repeated sampling and the use of Markov chains, where a Markov chain is a stochastic process in which every state can only depend on the state that precedes it. Let $x$ and $x'$ denote outcomes that can be interpreted as states of a Markov chain, where $x'$ precedes $x$. In MCMC methods, we use an iterative

procedure where we obtain $x'$ given $x$, update $x$ by the obtained value $x'$, and then repeat.

The Metropolis–Hastings (MH) algorithm [Has70] is a particular MCMC method for obtaining a sequence of observations from a probability distribution (known as the target distribution). We remark that the MH algorithm approximates the target distribution without the influence of autocorrelation, where autocorrelation refers to the case where subsequences of observations are correlated.

We proceed by describing the MH algorithm. Let $x$ and $x'$ denote states which are respective outcomes of random variables $X$ and $X'$. Let $g_{X' \mid X}$ and $g_{X \mid X'}$ denote conditional probability distributions; we refer to them as proposal distributions. Let $t \in \mathbb{N}$ be the number of iterations. In each iteration $i \in [t]$, the MH algorithm computes

$$p_i(x) = g_{X' \mid X}(x' \mid x) a(x, x'),$$

where

$$a(x, x') = \min\left(1, \frac{p_{i-1}(x')}{p_{i-1}(x)} \frac{g_{X \mid X'}(x \mid x')}{g_{X' \mid X}(x' \mid x)}\right)$$

is known as the acceptance rate and $p_0$ is fixed in advance.

When $i \in [t]$ increases, then $p_i$ approaches a stationary distribution and approximates more closely the target distribution. We remark that a stationary distribution is the probability distribution of a random variable in a stationary process, and a stationary process is a stochastic process where the unconditional joint probability distribution of the random variables does not change in time when the stochastic process is shifted in time.

## 2.5 Data privacy

In this section, we briefly review the field of data privacy. In particular, we review the threat model of honest-but-curious agents and the classic privatization mechanisms that are used in our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4).

At the beginning of this section, we discuss basic concepts of data privacy. In Subsection 2.5.1, we discuss a common measure of data privacy known as differential privacy.

For this section, we consult the survey by Aggarwal and Yu [AY08].

When speaking of data privacy, we tend do consider real-world communication networks of agents, where an agent is a user's device, a remote server, a computer cluster, etc. If the awareness of a particular attribute of an agent could lead to an unwanted profiling of the agent, we refer to such attribute as a sensitive attribute.

We briefly elaborate on the following data privacy measures:

- In $k$-anonymity, each sensitive attribute of each instance of a shared dataset has the same value as at least $k - 1$ respective sensitive attributes of other instances in the dataset.

- In $l$-diversity, each sensitive attribute of a shared dataset has at least $l$ distinct values that are present in it.
- In $(\epsilon, \delta)$-differential privacy, each sensitive attribute of a shared dataset is hidden under noise that is calibrated in a particular way based on parameters $\epsilon, \delta \geq 0$.

In this dissertation, we focus on differential privacy.

We proceed by discussing a distinction between local data privacy and central data privacy, we do so by partially consulting the tutorial slides by Nguyen [Ngu19].

In local data privacy, each agent shares its own dataset, and thus each agent is responsible on its own to make sure that the shared dataset meets the conditions of a selected data privacy measure. In central data privacy, each agent trusts a central curator, and thus it is the central curator who shares a dataset which contains instances (or statistics) of information provided by the agents. This way, it is the central curator who makes sure that the shared dataset meets the conditions of a selected data privacy measure.

We recall the threat model of honest-but-carious agents, where a threat model is a model that specifies what kind of access an attacker has to a system when attempting to disturb it or learn something that should be kept private.

Honest-but-curious agents [Gol04] is a threat model where the agents of a communication network follow the communication protocol (honesty) but they are also interested in discovering all available information (curiosity). We remark that honest-but-curious agents can be seen as a realistic threat model concerning local data privacy because, in local data privacy, the agents communicate among each other as opposed to relying on a central curator.

### 2.5.1 Differential privacy

In this subsection, we discuss differential privacy which is an actively studied measure of data privacy. In particular, we review the Gaussian mechanism and the Laplace mechanism which are used, respectively, in our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4).

For a review of differential privacy, we recommend and mostly follow the article by Dwork and Roth [DR14].

We start this subsection by recalling the definition of $(\epsilon, \delta)$-indistinguishability which was proposed by Dwork et al. [Dwo+06] and is commonly known as differential privacy.

**Definition 5.** *Let $\epsilon, \delta \geq 0$. A randomized algorithm $\mathcal{A}$ is $(\epsilon, \delta)$-differentially private if and only if, for all $\mathcal{S} \subseteq \text{image}(\mathcal{A})$ and for all tuples $(\mathcal{D}, \mathcal{D}')$ in a collection where adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$ differ only in the attributes of one instance, we have*

$$\Pr(\mathcal{A}(\mathcal{D}) \in \mathcal{S}) \leq e^{\epsilon} \Pr(\mathcal{A}(\mathcal{D}') \in \mathcal{S}) + \delta.$$

The aforementioned collection of tuples $(\mathcal{D}, \mathcal{D}')$ is a set of tuples of adjacent datasets, where adjacent datasets are datasets that differ in one row, and where a dataset is a table whose rows are over instances and the columns are over attributes.

We highlight that, in local differential privacy, each agent adds noise to its sensitive attributes on its own, whereas, in central differential privacy, a central curator typically adds noise to statistics computed from attributes. Thus, in central differential privacy, it is more common to refer to sensitive statistics as opposed to sensitive attributes. We remark that adding noise to sensitive statistics as opposed to sensitive attributes results in a lower error between a privatized statistic and a sensitive statistic, when such statistics are averages computed from attributes.

Regarding the motivation of central differential privacy, bounding the probability of observing one adjacent dataset by the probability of observing another adjacent dataset (as in Definition 5) can be interpreted as a level of privatization, which is measured by the privacy budget $(\epsilon, \delta)$, against a reconstruction attack, where by a reconstruction attack we refer to an identification of the value of a sensitive attribute. Referring to Garfinkel et al. [GAM19], we remark that, in a reconstruction attack, an attacker can reconstruct sensitive attributes by making comparisons of the following two statistics: a statistic that is computed over the sensitive attributes and a statistic that is computed over the same sensitive attributes except some of them being removed or modified in a particular way.

We remark that, when the privacy budget $(\epsilon, \delta)$ is higher, the noisiness of privatized statistics or privatized attributes is lower because it is easier to meet the bound in Definition 5.

We proceed by recalling definitions of the Gaussian mechanism and the Laplace mechanism, which are classic privatization mechanisms for adding noise to sensitive attributes.

Firstly, we define the Gaussian mechanism.

**Definition 6.** *Let $\epsilon, \delta > 0$. Let $\mathcal{X} \subseteq \mathbb{R}^n$, where $n \in \mathbb{N}$. Let $f : \mathcal{X} \to \mathbb{R}^k$, where $k \in \mathbb{N}$. The Gaussian mechanism is a mechanism that privatizes a value $f(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{X}$, by adding to it independent observations of the following Gaussian random variable:*

$$\mathcal{N}\left(0, \frac{2\log(1.25/\delta)\Delta_2^2(f)}{\epsilon^2}\right),$$

*where the $l_2$ sensitivity*

$$\Delta_2(f) = \max_{\text{adjacent } \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}} \|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)\|_2. \tag{2.13}$$

We remark that generation of noisy values according to the Gaussian mechanism is a classic strategy to guarantee $(\epsilon, \delta)$-differential privacy.

Secondly, we define the Laplace mechanism.

**Definition 7.** *Let $\epsilon > 0$. Let $\mathcal{X} \subseteq \mathbb{R}^n$, where $n \in \mathbb{N}$. Let $f : \mathcal{X} \to \mathbb{R}^k$, where $k \in \mathbb{N}$. The Laplace mechanism is a mechanism that privatizes a value $f(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{X}$, by adding to it independent observations of the following Laplace random variable:*

$$\text{lap}\left(0, \frac{\Delta(f)}{\epsilon}\right),$$

*where the $l_1$ sensitivity*

$$\Delta(f) = \max_{\text{adjacent } \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}} \|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)\|_1. \tag{2.14}$$

We remark that generation of noisy values according to the Laplace mechanism is a classic strategy to guarantee $(\epsilon, 0)$-differential privacy (also known as $\epsilon$-differential privacy).

Finally, we highlight that Dwork and Roth [DR14] provides a composition theorem which states that a collection of size $k$ of $(\epsilon, \delta)$-differentially private randomized algorithms is $(k\epsilon, k\delta)$-differentially private. In our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4) we reuse the aforementioned composition theorem. In particular, we refer to $(k\epsilon, k\delta)$ as the (total) privacy budget and we refer to $(\epsilon, \delta)$ as an even split (one of $k$ parts) of the total privacy budget.

## 2.6 Mathematical optimization

In this section, we briefly review mathematical optimization. In particular, we review the standard form of a convex program, which is accepted by the `cvxopt` package, and that is used in our contribution on the tailored noise mechanism (discussed in Chapter 4).

At the beginning of this section, we discuss basic vocabulary of mathematical optimization and convex programs. In Subsection 2.6.1, we discuss the basic principles of implementing a convex program using the `cvxopt` package.

For a review of mathematical optimization, we recommend and mostly follow the textbook by Boyd and Vandenberghe [BV14].

We start by reviewing mathematical optimization.

Mathematical optimization is a field on solving optimization problems that can be stated in the following form: we intend to

| | |
|---|---|
| minimize | $f_0(\mathbf{w})$ |
| subject to the constraints | $f_1(\mathbf{w}) \leq b_1, f_2(\mathbf{w}) \leq b_2, \ldots, f_k(\mathbf{w}) \leq b_k,$ |

where $\mathbf{w} \in \mathbb{R}^n$ are optimization variables (also known as constraint variables), $n \in \mathbb{N}$, $f_0 : \mathbb{R}^n \to \mathbb{R}$ is an objective function, $f_1, f_2, \ldots, f_k : \mathbb{R}^n \to \mathbb{R}$ are constraint functions, $b_1, b_2, \ldots, b_k \in \mathbb{R}$ are upper-bounds of constraint functions, and $k \in \mathbb{N}$ is the number of constraints.

An optimization problem is feasible if there exist values of the constraint variables so that the constraints of the optimization problem are met. Otherwise, an optimization problem is infeasible. For example, let $w_1$ and $w_2$ be constraint variables whose values are in the interval $[1, 2]$. If an optimization problem consists of the equality constraint $w_1 + w_2 \leq 2$, then the optimization problem is feasible because $w_1$ and $w_2$ can take the value 1. However, upon the inequality constraint $w_1 + w_2 \leq 1$, the optimization problem is infeasible because $w_1 + w_2$ is at least 2.

If $\mathbf{w}$ results in the lowest value of the objective function of an optimization problem such that the constraints are met, then we refer to $\mathbf{w}$ as a solution of the optimization problem. We remark that a value of the objective function is referred to as an objective value.

We briefly elaborate on some subfields of mathematical optimization:

- In linear optimization, the objective function and the constraint functions are linear functions.
- In non-linear optimization, the objective function and the constraint functions are non-linear functions.
- In convex optimization, the objective function and the constraint functions are convex functions, where a function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if the domain of $f$ is a convex set $S$ and if for each $x, y \in S$, we have $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$, where $n \in \mathbb{N}$ and $0 \leq \alpha \leq 1$ (a set $S$ is convex if, for any $x, y \in S$ and any $0 \leq \alpha \leq 1$ , we have $\alpha x + (1-\alpha)y \in S$). This way, convex optimization is a generalization of linear optimization or a special case of non-linear optimization.

In this dissertation, we focus on convex optimization.

We highlight that optimization problems solved in convex optimization can be stated in the following form: we intend to

$$\begin{array}{ll} \text{minimize} & f_0(\mathbf{w}) \\ \text{subject to the constraints} & f_1(\mathbf{w}) \leq 0, f_2(\mathbf{w}) \leq 0, \ldots, f_k(\mathbf{w}) \leq 0, \\ & g_1(\mathbf{w}) = 0, g_2(\mathbf{w}) = 0, \ldots, g_l(\mathbf{w}) = 0, \end{array}$$

where $\mathbf{w} \in \mathbb{R}^n$ are constraint variables, $n \in \mathbb{N}$, $f_0 : \mathbb{R}^n \to \mathbb{R}$ is a (convex) objective function, $f_1, f_2, \ldots, f_k : \mathbb{R}^n \to \mathbb{R}$ are (convex) inequality constraint functions, $k \in \mathbb{N}$ is the number of inequality constraints, $g_1, g_2, \ldots, g_l : \mathbb{R}^n \to \mathbb{R}$ are (convex) equality constraint functions, and $l \in \mathbb{N}$ is the number of equality constraints. We refer to an optimization problem solved in convex optimization as a convex program.

Finally, we remark that, in this dissertation, we consider algorithms that are designed to find solutions of convex programs in an iterative manner and under the stopping condition which is met when an infeasibility measure gets lower than a prescribed tolerance (i.e., in such algorithms, we allow for some slack that concerns the feasibility). We refer to such algorithms as constraint solvers. To summarize, we consider constraint solvers that are guided by the following two factors:

- The feasibility of the constraints.
- The minimization of the objective function.

### 2.6.1 The `cvxopt` package

In this subsection, we discuss the basic principles of solving a convex program using `cvxopt` [ADV20] which is a python library. We remark that `cvxopt` is used in our contribution on the tailored noise mechanism (discussed in Chapter 4)

We describe the elements of a convex program, which are accepted by the standard form of `cvxopt`. Let $\mathbf{w} \in \mathbb{R}^n$ be the constraint variables of a convex program, where $n$ is the number of constraint variables. The arguments to the convex constraint solver of `cvxopt` are data structures that represent the linear equality constraints, the linear inequality constraints, the objective function, and the non-linear constraints of a convex program:

- The linear equality constraints. Let $\mathbf{A} \in \mathbb{R}^{k \times n}$ be a matrix whose elements are the coefficients of constraint variables in the linear equality constraints, and where $k$ is the number of linear equality constraints. Let $\mathbf{b} \in \mathbb{R}^k$ be a vector whose elements are the constant terms in the linear equality constraints. The aim of the constraint solver is to find such $\mathbf{w}$ that $\mathbf{A}\mathbf{w} = \mathbf{b}$.

- The linear inequality constraints. Let $\mathbf{G} \in \mathbb{R}^{l \times n}$ be a matrix whose elements are the coefficients of constraint variables in the linear inequality constraints, and where $l$ is the number of linear inequality constraints. Let $\mathbf{h} \in \mathbb{R}^l$ be a vector whose elements are the constant terms in the linear inequality constraints. The aim of the constraint solver is to find such $\mathbf{w}$ that $\mathbf{G}\mathbf{w} \preceq \mathbf{h}$ (where $\preceq$ denotes "element-wise less than or equal").

- The objective function. Let $f_0^{\text{cvx}} : \mathbb{R}^n \to \mathbb{R}^+$ denote a twice-differentiable objective and convex function, where $\mathbb{R}^+$ is the space of non-negative real numbers. The aim of the constraint solver is to minimize $f_0^{\text{cvx}}(\mathbf{w})$.

- The non-linear constraints. Let $f_1^{\text{cvx}}, f_2^{\text{cvx}}, \ldots, f_m^{\text{cvx}} : \mathbb{R}^n \to \mathbb{R}$ denote twice-differentiable non-linear (yet convex) constraint functions, where $m$ is the number of non-linear constraints. The aim of the constraint solver is to find such $\mathbf{w}$ that, for each $i \in [m]$, we have $f_i^{\text{cvx}} \preceq \mathbf{0}$.

In each iteration of the constraint solver, `cvxopt` requires to evaluate the first-order partial derivatives and the second-order partial derivatives of the objective function $f_0^{\text{cvx}}$ and the non-linear constraint functions $f_1^{\text{cvx}}, f_2^{\text{cvx}}, \ldots, f_m^{\text{cvx}}$.

The first-order partial derivatives are stored in a matrix $\mathbf{D} \in \mathbb{R}^{(1+m) \times n}$ whose first row contains the first-order partial derivatives of $f_0^{\text{cvx}}$ evaluated on $\mathbf{w}$, and whose subsequent rows contain the first-order partial derivatives of non-linear constraints $f_1^{\text{cvx}}, f_2^{\text{cvx}}, \ldots, f_m^{\text{cvx}}$ evaluated on $\mathbf{w}$.

The second-order partial derivatives are stored in a matrix $\mathbf{H} \in \mathbb{R}^{n \times n}$ such that

$$\mathbf{H} = \sum_{i=\{0\} \cup [m]} z_i \nabla^2 f_i^{\text{cvx}}(\mathbf{w}),$$

where $\nabla^2 f_i^{\text{cvx}}(\mathbf{w})$ is the matrix of the second-order partial derivatives of $f_i^{\text{cvx}}$ evaluated on $\mathbf{w}$, and $z_0, z_1, \ldots, z_{n^m}$ are scalars provided by the constraint solver.

We highlight that the matrix $\mathbf{A}$ of linear equality constraints, the matrix $\mathbf{G}$ of linear inequality constraints, the matrix $\mathbf{D}$ of first-order partial derivatives, and the matrix $H$ of second-order partial derivatives can be sparse, i.e., most of their entries equal to 0. Storing those matrices in data structures dedicated to sparse matrices reduces the computational cost and the memory cost of the convex program.

## 2.7 Summary

In this section, we briefly summarize how the fields reviewed in this chapter make part of the settings that are related to this dissertation. First, we provide a diagram that indicates the relations among the fields and the settings. Then, we briefly

elaborate on the settings. Finally, we summarize the main reasons why the reviewed fields are selected for this dissertation.

In Figure 2.5, we illustrate the relation among the fields and the settings that are part of (or close to) the scope of this dissertation. We remark that we use simplified and broadened naming of the settings because our main intention is to highlight the boundary between the aspects that are part and outside the scope of this dissertation.

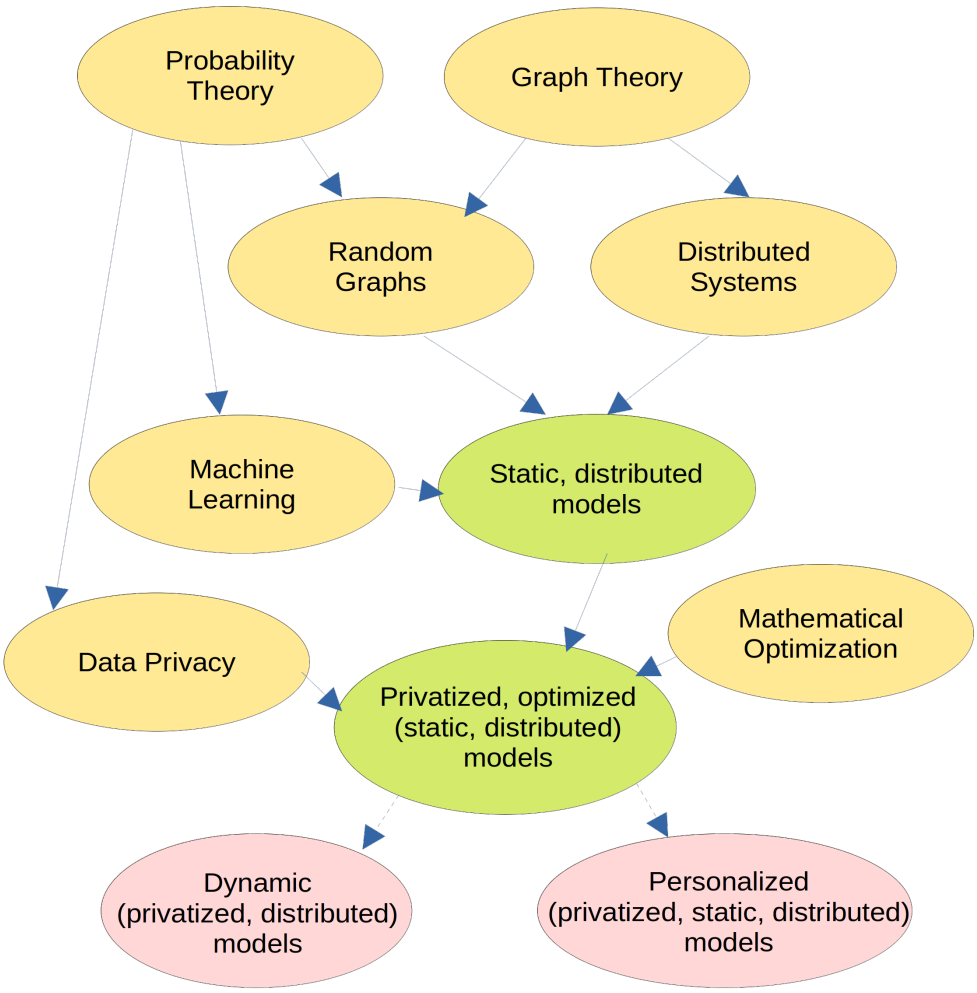


Figure 2.5: Relation among the fields and the settings considered in the dissertation. The arrows indicate that a field (or a setting) influences another field (or another setting). The ovals at the bottom (pointed at by dashed arrows) are settings that are outside (yet close to) the scope of this dissertation. The ovals at the center are settings that are part of the scope.

We briefly elaborate on the settings (illustrated in Figure 2.5) that are part of (or close to) the scope of this dissertation.

The setting with static, distributed models refers to the case when a communication network of agents is modeled by a fixed graph, whereas the setting with dynamic models (outside the scope) refers to the case where the graph of the

communication network changes over time.

The setting with optimized, privatized models refers to the case with utility-maximizing privatization mechanisms, as opposed to the classic privatization mechanisms, e.g., the Gaussian mechanism (Definition 6) and the Laplace mechanism (Definition 7). We remark that classic privatization mechanisms result, typically, in more privatization noise than necessary (discussed in Chapter 4).

The setting with personalized models (outside the scope) refers to the case where each agent (or a distinct subgroup of agents) of a communication network aims for learning local statistical models, as opposed to the case where all the agents aim for learning the same statistical model.

We summarize the main reasons why the reviewed fields (illustrated in Figure 2.5) are selected for this dissertation.

Regarding probability theory, we use Hoeffding's inequality in our example of distributed averaging on Erdős–Rényi graphs (discussed in Section 3.2) and we use U-statistics in our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4).

Regarding graph theory, we use power-law degree sequences in our contribution on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3).

Regarding random graphs, we use the Erdős–Rényi model and the configuration model, respectively, in our example and contribution on distributed averaging on random graphs (discussed, respectively, in Section 3.2 and Section 3.3).

Regarding distributed systems, we discuss overlay networks and gossip algorithms for distributed averaging in our examples and contribution on the bias in distributed averaging when the graph of agents is non-regular (discussed in Chapter 3). Also, we discuss a central curator which is present in our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4).

Regarding machine learning, we use linear regression models in our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4).

Regarding data privacy, we use the threat model of honest-but-curious agents and the classic privatization mechanisms in our contributions on distributed averaging on graphs with arbitrary degree sequences (discussed in Section 3.3) and the tailored noise mechanism (discussed in Chapter 4).

Finally, regarding mathematical optimization, we review the standard form of a convex program, which is accepted by the `cvxopt` package, and that is used in our contribution on the tailored noise mechanism (discussed in Chapter 4). In particular, we use a convex program to design a utility-maximizing privatization mechanism.

# Chapter 3

# Bias in distributed averaging

In this chapter, we study the setting of distributed averaging (discussed in Subsection 2.3.2), where each agent of a communication network is attributed a vector of individual values, and where each agent intends to collaboratively (without a central curator) compute the unbiased averages over all the aforementioned vectors. In particular, we aim for performing distributed averaging while mitigating the bias which is present when the graph of agents is non-regular (i.e., the numbers of neighbors vary).

Usually, existing works solve this problem by assuming that either (i) each agent reveals its degree to its neighbors or (ii) every two neighboring agents can perform handshakes (discussed in Section 2.3) in every exchange of information. However, the degrees reveal information about the profiles of the agents and the handshakes are impractical upon inactive agents.

**Outline.** In Section 3.1, we discuss two approaches for computing an unbiased empirical distribution of the individual values attributed to the agents: the first approach is based on overlay networks (discussed in Section 2.3) and the second approach is based on the simple gossip algorithm (discussed in Subsection 2.3.1). In Section 3.2, we walk through an example of proving an asymptotic guarantee for the spectral norm of the difference between the linear transformation $\frac{\mathbf{1}^{\mathrm{T}}\mathbf{1}}{n}$ (centralized averaging) and the linear transformation due to the simple gossip algorithm upon convergence, where $\mathbf{1}$ is the vector of 1's of length $n$, and when the graph of agents is modeled by the ER model (Definition 2). In Section 3.3, we prove an asymptotic guarantee for the mean squared error between the average of sensitive attributes and the average of locally privatized attributes computed by a bias-correcting variant of the simple gossip algorithm, when the graph of agents is modeled by the configuration model (Definition 4), and when the agents interact without handshakes.

## 3.1 Estimating empirical distributions

At the beginning of this section, we formulate the problem of estimating a probability distribution in a distributed manner.

Let $G = (V, E, \lambda, \Sigma)$ be a vertex-labeled graph, where $\Sigma$ is a set of (all possible) labels (real-valued vectors) and a function $\lambda : V \to \Sigma$ assigns a label to a vertex.

Our communication network is on the graph $G$, and each agent (a vertex in $V$) is assigned a label $l$ which is drawn independently from some fixed but unknown probability distribution on $\Sigma$.

We aim for approximating the unknown probability distribution by an empirical distribution $\hat{p}_\Sigma$, where the probability of observing a label $l$ is the following:

$$\hat{p}_\Sigma(l) = \frac{|\{v \in V \mid \lambda(v) = l\}|}{|V|}. \tag{3.1}$$

In Subsection 3.1.1, we discuss computation of an unbiased empirical distribution by an overlay network (discussed in Section 2.3). In Subsection 3.1.2, we discuss computation of an unbiased empirical distribution using gossip algorithms (discussed in Subsection 2.3.1).

### 3.1.1 Approach with an overlay network

The agents of a communication network can obtain the empirical distribution $\hat{p}_\Sigma$ (Equation 3.1) in a distributed manner using an overlay network which is a tree.

We summarize the application of such overlay network as follows:

1. The agents compute a spanning tree of the graph, for example, by an algorithm discussed in Gallager et al. [GHS83]

2. The agents propagate their labels (and all the labels they receive from others during the process) towards the root of the spanning tree. (If a smaller sample is desired they toss a coin and only propagate their own label with the probability equal to the size of the desired sample divided by the number of vertices). Since the overlay network is a tree, the root receives each label only once, and thus the computation of the empirical distribution $\hat{p}_\Sigma$ at the root is unbiased.

3. The root broadcasts the sample back through the spanning tree overlay network, thus each agent is aware of the empirical distribution $\hat{p}_\Sigma$.

The issue with the approach of computing the empirical distribution $\hat{p}_\Sigma$ by a spanning tree overlay network is that the transmission of the final sample is not robust upon failure or inactivity of a single agent. It is even worse when the failing agent is closer to the root because less agents in total can receive the result.

### 3.1.2 Approach with gossip algorithms

The agents of a communication network can compute the empirical distribution $\hat{p}_\Sigma$ (Equation 3.1) by distributed averaging using gossip algorithms.

In this subsection, we firstly recall the simple gossip algorithm (Algorithm 1) and illustrate an example of applying Algorithm 1 for estimating the empirical distribution $\hat{p}_\Sigma$ (we do so by defining Algorithm 2). Then, we show that we have a bias between the empirical distribution estimated by Algorithm 2 and the empirical distribution $\hat{p}_\Sigma$, when the graph $G = (V, E, \lambda, \Sigma)$ of the communication network is non-regular (when the vertices have unequal degrees). Finally, we recall how the

Metropolis–Hastings step (discussed in Subsection 2.3.2) can improve Algorithm 2, so that the aforementioned bias is corrected when each agent makes use of the degrees of its neighbors.

**Simple gossip algorithm.** We recall the simple gossip algorithm by defining Algorithm 1 below, which is a special case of gossip algorithms discussed by Boyd et al. [Boy+06] We remark that one iteration of Algorithm 1 is a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^n$, i.e., the transition matrix $\mathbf{T} = \mathrm{diag}(\mathbf{d})^{-1}\mathbf{A}$, where $\mathbf{A}$ is the adjacency matrix of the graph $G$ of agents and $\mathbf{d}$ is the degree sequence of $G$. In other words, one iteration of Algorithm 1 involves each agent averaging the values revealed by its neighbors. This way, the execution of Algorithm 2 for the number $\mathrm{t_{go}}$ of gossip iterations results in the linear transformation $\mathbf{T}^{\mathrm{t_{go}}}$.

**Algorithm 1:** `SimpleGossip` (`SiGo`)

**Input :** $\mathbf{A} \in [0,1]^{n\times n}$: adjacency matrix of the graph $G$ of agents
$\mathrm{t_{go}} \in \mathbb{N}$ : number of gossip iterations
$\mathbf{w} \in \mathbb{R}^n$
**Output :** $\mathbf{z} \in \mathbb{R}^n$

$\mathbf{d} \leftarrow \sum_{i\in[n]} \mathbf{A}_{:,i}$
$\mathbf{T} \leftarrow \mathrm{diag}(\mathbf{d})^{-1}\mathbf{A}$
$\mathbf{z} \leftarrow \mathbf{T}^{\mathrm{t_{go}}}\mathbf{w}$

We illustrate an example of applying `SiGo` (Algorithm 1) for computing the order $n$ of $G$, when $G$ is a complete graph and has all self-loops. We assign a label 1 to one agent, and a label 0 to the remaining $n-1$ agents. In the 1-st iteration of `SiGo`, each agent computes the average $\frac{1+0+\ldots+0}{n} = \frac{1}{n}$. By taking the reciprocal of $\frac{1}{n}$, each agent computes $n$.

We proceed by illustrating an example of applying Algorithm 1 for estimating the empirical distribution $\hat{p}_\Sigma$, we do so by defining Algorithm 2 below. Firstly, for each vertex $v \in V$ with a label $\lambda(v)$, let $\mathbb{I}[\lambda(v)] \in \{0,1\}^{|\Sigma|}$ be the initial estimate of the empirical distribution $\hat{p}_\Sigma$, where the probability of the label $\lambda(v)$ is 1 and all other probabilities are 0. Then, we remark that the output of Algorithm 2 is a matrix, as opposed to a vector (as in Algorithm 1). This way, for $i \in [n]$, the $i$-th row of the aforementioned matrix contains an estimate of the empirical distribution $\hat{p}_\Sigma$ computed by the agent $i$. Also, we remark Algorithm 1 corresponds to centralized simulation of distributed averaging, whereas Algorithm 2 corresponds to a decentralized one.

**Algorithm 2:** `SimpleGossip` for estimating a distribution

**Input :** $G = (V, E, \lambda, \Sigma)$ : labeled graph of order $n$
$\mathrm{t_{go}} \in \mathbb{N}$ : number of gossip iterations
**Output :** $\mathrm{Y}^{(\mathrm{t_{go}})} \in \mathbb{R}^{n\times|\Sigma|}$ : estimates of the empirical distribution $\hat{p}_\Sigma$

**for** $v \in V$ **do**
  $\mathrm{Y}^{(0)}_{v,:} \leftarrow \mathbb{I}[\lambda(v)]$
**for** $j \leftarrow 1$ **to** $\mathrm{t_{go}}$ **do**
  **for** $v \in V$ **do**
    $\mathrm{Y}^{(j)}_{v,:} \leftarrow \frac{\sum_{u\in N(v)} \mathrm{Y}^{(j-1)}_{u,:}}{\deg(v)}$

In Line 6 of Algorithm 2, each agent iteratively updates its estimate by averaging the estimates computed by its neighbors. This way, when $G$ is non-regular, there exist such $v, u \in V$ so that $\deg(v) \neq \deg(u)$. Thus, when Algorithm 2 converges, we have a bias (also discussed in Subsection 2.3.2) between a value of the empirical distribution estimated by Algorithm 2 and the respective value of the empirical distribution $\hat{p}_\Sigma$, when $G$ is non-regular.

**Metropolis–Hastings gossip algorithm.** Algorithm 2 can be improved using the Metropolis–Hastings step (discussed in Subsection 2.3.2) in order to obtain unbiased estimates of the values of the empirical distribution $\hat{p}_\Sigma$, when $G$ is non-regular; such improvement is defined by Algorithm 3 where for any neighboring vertices $u, v \in V$, we have $f_{\mathrm{MH}}(u, v) = 1$ if $\deg(v) \geq \deg(u)$ and $f_{\mathrm{MH}}(u, v) = \frac{\deg(v)}{\deg(u)}$ otherwise. We remark that the simple gossip algorithm with the Metropolis–Hastings step (`MetropolisHastingsGossip`) is discussed in Kenyeres and Kenyeres [KK20] as well as Giaretta and Girdzijauskas [GG19].

**Algorithm 3:** `MetropolisHastingsGossip` for estimating a distribution

**Input :** $G = (V, E, \lambda, \Sigma)$ : labeled graph of order $n$
$\mathrm{t_{go}} \in \mathbb{N}$ : number of gossip iterations
**Output :** $\mathrm{Y}^{(\mathrm{t_{go}})} \in \mathbb{R}^{n \times |\Sigma|}$ : estimates of the empirical distribution $\hat{p}_\Sigma$

1 **for** $v \in V$ **do**
2 $\quad \mathrm{Y}^{(0)}_{v,:} \leftarrow \mathbb{I}[\lambda(v)]$
3 **for** $j \leftarrow 1$ **to** $\mathrm{t_{go}}$ **do**
4 $\quad$ **for** $v \in V$ **do**
5 $\quad\quad \mathrm{Y}^{(j)}_{v,:} \leftarrow \frac{\sum_{u \in N(v)} \left( f_{\mathrm{MH}}(u,v) \mathrm{Y}^{(j-1)}_{u,:} + \left(1 - f_{\mathrm{MH}}(u,v)\right) \mathrm{Y}^{(j-1)}_{v,:} \right)}{\deg(v)}$

When Algorithm 3 converges, each value of the empirical distribution $\hat{p}_\Sigma$ (present in each row of the output) is unbiased due to the factor $f_{\mathrm{MH}}(u, v)$ and the factor $\left(1 - f_{\mathrm{MH}}(u, v)\right)$ in Line 6. We remark that, for computing $f_{\mathrm{MH}}(u, v)$, each agent has to be aware of the degree of its neighbors, while in some contexts the degree is a sensitive attribute.

We recall that Algorithm 1, Algorithm 2, and Algorithm 3 converge when the graph $G$ is connected and has a self-loop or a cycle of odd length, as discussed in Subsection 2.3.2.

## 3.2 Averaging on ER graphs

In this section, we continue to study the bias between the average of individual values computed by the simple gossip algorithm (Algorithm 1) and the true average of individual values, when the graph $G = (V, E)$ of agents is non-regular, as initially discussed in Subsection 3.1.2. When $G$ is regular, the linear transformation due to the simple gossip algorithm upon convergence equals the linear transformation $\frac{\mathbf{1}^{\mathrm{T}}\mathbf{1}}{n}$ (centralized averaging), thus the bias is absent, where $\mathbf{1}$ is the vector of 1's of length $n = |V|$.

In particular, we walk through an example of proving an asymptotic guarantee for the spectral norm of the difference between the linear transformation $\frac{\mathbf{1}^{\mathrm{T}}\mathbf{1}}{n}$ and

the linear transformation due to the simple gossip algorithm (Algorithm 1) upon convergence, when the graph of agents is modeled by the ER model (Definition 2). Instead of a scalar transition matrix $\mathbf{T}$, we consider a random transition matrix $\boldsymbol{T} = \operatorname{diag}(\boldsymbol{D})^{-1}\boldsymbol{A}$, where $\boldsymbol{A}$ is the random adjacency matrix of a random graph modeled by the ER model, and $\boldsymbol{D}$ is the random degree sequence of the random graph. We remark that the degree sequence $\boldsymbol{D}$ can be computed from the rows of (and thus depends on) the adjacency matrix $\boldsymbol{A}$.

We intend to quantify the difference between the linear transformation $\frac{\mathbf{1}^{\mathrm{T}}\mathbf{1}}{n}$ (centralized averaging) and the linear transformation due to the simple gossip algorithm (Algorithm 1) upon convergence, when the simple gossip algorithm is executed on a graph modeled by the ER model. Thus, we intend to upper-bound the spectral norm $\left\| \boldsymbol{T}^{\mathrm{t_{go}}} - \frac{\mathbf{1}^{\mathrm{T}}\mathbf{1}}{n} \right\|_2$, where $\mathrm{t_{go}}$ is the number of gossip iterations.

In particular, we provide Theorem 1 and walk through its proof. We remark that some techniques used in the proof of Theorem 1 result in a suboptimal asymptotic guarantee compared to similar works such as Lugosi et al. [LMZ18] However, by Theorem 1, we intend to provide an upper bound for the spectral norm $\left\| \boldsymbol{T}^{\mathrm{t_{go}}} - \frac{\mathbf{1}^{\mathrm{T}}\mathbf{1}}{n} \right\|_2$ instead of the spectral norm $\left\| \frac{1}{np}\boldsymbol{A} - \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n} \right\|_2$, where the latter spectral norm is found in the literature, and where, for the former spectral norm, we have not found a reference in the literature. Also, we consider a transition matrix $\boldsymbol{T}$ which is random as opposed to fixed, and remark that the case where such transition matrix is fixed is discussed in the slides by Iutzeler [Iut16a].

**Theorem 1.** *Let $\boldsymbol{A}$ be the random adjacency matrix of an ER random graph modeled by the probability mass function $p_G(G \mid n, p)$, where $n$ is the order of the graph, $p$ is the probability of edge assignment between any two distinct vertices of the ER random graph, and $G$ is an observation of the ER random graph. Let $\boldsymbol{D}$ be the random degree sequence of the ER random graph. Let $\boldsymbol{T} = diag(\boldsymbol{D})^{-1}\boldsymbol{A}$. Let us assume that the 2-nd largest singular value of $\boldsymbol{T}$ is $\mathcal{O}\left(\frac{\log n}{\sqrt{n}}\right)$. (We support the assumption in the proof.) With high probability, we have*

$$\lim_{\mathrm{t_{go}} \to \infty} \left\| \boldsymbol{T}^{\mathrm{t_{go}}} - \frac{\mathbf{1}^{\mathrm{T}}\mathbf{1}}{n} \right\|_2 = \mathcal{O}\left(\frac{\log n}{\sqrt{n}}\right),$$

*when $p \in \left[\frac{(\log n)^2}{n}, 1 - \frac{(\log n)^2}{n}\right]$.*

*Proof.* Our proof is split in five parts. In the first part, we decompose the spectral norm in three components. From the second to the fourth part, we upper bound, respectively, the spectral norm of each of the decomposed components. In the fifth part, we compose the three upper bounds to obtain our result.

Decomposition. We decompose the spectral norm as follows:

$$\begin{aligned}
\left\| \boldsymbol{T}^{\mathrm{t_{go}}} - \frac{\mathbf{1}^{\mathrm{T}}\mathbf{1}}{n} \right\|_2 &= \left\| \boldsymbol{T}^{\mathrm{t_{go}}} - \boldsymbol{T} + \boldsymbol{T} - \frac{1}{np}\boldsymbol{A} + \frac{1}{np}\boldsymbol{A} - \frac{\mathbf{1}^{\mathrm{T}}\mathbf{1}}{n} \right\|_2 \\
&\leq \left\| \boldsymbol{T}^{\mathrm{t_{go}}} - \boldsymbol{T} \right\|_2 + \left\| \boldsymbol{T} - \frac{1}{np}\boldsymbol{A} \right\|_2 + \left\| \frac{1}{np}\boldsymbol{A} - \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n} \right\|_2,
\end{aligned}$$

where the second line is due to the triangle inequality.

The first component $\left\| \boldsymbol{T}^{t_{go}} - \boldsymbol{T} \right\|_2$. Let $\boldsymbol{U}$ be the random matrix of eigenvectors of the random transition matrix $\boldsymbol{T}$, and let $\boldsymbol{V}$ be the random diagonal matrix of eigenvalues of $\boldsymbol{T}$, so that $\boldsymbol{U}\boldsymbol{V}\boldsymbol{U}^{-1}$ is the eigendecomposition of $\boldsymbol{T}$.

We highlight that

$$\begin{aligned} \boldsymbol{T}^{t_{go}} &= \boldsymbol{U}\boldsymbol{V}\boldsymbol{U}^{-1} \dots \boldsymbol{U}\boldsymbol{V}\boldsymbol{U}^{-1} \\ &= \boldsymbol{U}\boldsymbol{V}^{t_{go}}\boldsymbol{U}^{-1}. \end{aligned}$$

This way,

$$\begin{aligned} \left\| \boldsymbol{T}^{t_{go}} - \boldsymbol{T} \right\|_2 &= \left\| \boldsymbol{U}\boldsymbol{V}^{t_{go}}\boldsymbol{U}^{-1} - \boldsymbol{U}\boldsymbol{V}\boldsymbol{U}^{-1} \right\|_2 \\ &= \left\| \boldsymbol{U}\left(\boldsymbol{V}^{t_{go}} - \boldsymbol{V}\right)\boldsymbol{U}^{-1} \right\|_2. \end{aligned}$$

We recall that the spectral norm of a matrix equals the largest singular value of the matrix (indicated by Equation 2.1), and singular values can be obtained from respective eigenvalues. Then, we recall that the eigenvalues of the matrix $\boldsymbol{U}\left(\boldsymbol{V}^{t_{go}} - \boldsymbol{V}\right)\boldsymbol{U}^{-1}$ are in the diagonal of the matrix $\boldsymbol{V}^{t_{go}} - \boldsymbol{V}$, since the matrix $\boldsymbol{U}\left(\boldsymbol{V}^{t_{go}} - \boldsymbol{V}\right)\boldsymbol{U}^{-1}$ is the eigendecomposition of the matrix $\boldsymbol{T}^{t_{go}} - \boldsymbol{T}$.

An observation of $\boldsymbol{T}$ is a row stochastic matrix because the sum of each row of $\boldsymbol{T}$ is constrained to equal 1. In Banerjee and Mehatari [BM16], the authors indicate that 1 is the highest singular value of a row stochastic matrix. This way, $(1, \mathbf{1})$ is a pair of an eigenvalue and an eigenvector of $\boldsymbol{T}$. In Paz [Paz63], the author indicates that $\lim_{t_{go}\to\infty} \boldsymbol{T}^{t_{go}}$ exists when $G$ is connected and has a self-loop or a cycle of odd length. Then, in Frieze and Karoński [FK16], the authors show that $G$ is connected, with high probability, when $n \to \infty$ and $p \in \left[\frac{\log n}{n}, 1\right]$. Furthermore, we remark that a triangle graph is a cycle of odd length, and, referring to Gilmer and Kopparty [GK16], we remark that the probability of the existence of a triangle subgraph between three vertices of an ER graph is $p^3$. This way, with high probability, $G$ has a cycle of odd length, and the only non-zero eigenvalue in $\lim_{t_{go}\to\infty} \boldsymbol{V}^{t_{go}}$ is 1 (we recall that the diagonal of $\lim_{t_{go}\to\infty} \boldsymbol{V}^{t_{go}}$ contains the eigenvalues of $\lim_{t_{go}\to\infty} \boldsymbol{T}^{t_{go}}$). As a result, the respective eigenvalue in $\lim_{t_{go}\to\infty} \boldsymbol{V}^{t_{go}} - \boldsymbol{V}$ is $1 - 1 = 0$.

We assume that the 2-nd largest singular value of $\boldsymbol{T}$ is, with high probability, $\mathcal{O}\left(\frac{\log n}{\sqrt{n}}\right)$, when $p \in \left[\frac{(\log n)^2}{n}, 1 - \frac{(\log n)^2}{n}\right]$. We support the assumption by firstly indicating that Frieze and Karoński [FK16] show that, with high probability, the 2-nd largest singular value of the adjacency matrix of an ER random graph is $\mathcal{O}(\sqrt{n}\log n)$, when $p \in \left[\frac{(\log n)^2}{n}, 1 - \frac{(\log n)^2}{n}\right]$. (We remark that we were unable to interpret and quantify exactly the high probability of the aforementioned argument, thus leaving it as a future direction.) Secondly, since the expected degree of a vertex of an ER random graph is $(n-1)p$, we indicate that every element of $\text{diag}(\boldsymbol{D})^{-1}$ is $\mathcal{O}(\frac{1}{n})$. Since $\text{diag}(\boldsymbol{D})^{-1}$ is a diagonal matrix, the second largest singular value of $\boldsymbol{T}$ is $\mathcal{O}(\sqrt{n}\log n)\mathcal{O}\left(\frac{1}{n}\right) = \mathcal{O}\left(\frac{\log n}{\sqrt{n}}\right)$.

Hence, with high probability,

$$\lim_{t_{go}\to\infty} \left\| \boldsymbol{T}^{t_{go}} - \boldsymbol{T} \right\|_2 = \mathcal{O}\left(\frac{\log n}{\sqrt{n}}\right), \tag{3.2}$$

when $p \in \left[\frac{(\log n)^2}{n}, 1 - \frac{(\log n)^2}{n}\right]$.

The second component $\left\| \boldsymbol{T} - \frac{1}{np}\boldsymbol{A} \right\|_2$. We perform the following rearrangement:

$$\begin{aligned}
\left\| \boldsymbol{T} - \frac{1}{np}\boldsymbol{A} \right\|_2 &= \left\| \mathrm{diag}(\boldsymbol{D})^{-1}\boldsymbol{A} - \frac{1}{np}\boldsymbol{A} \right\|_2 \\
&= \left\| \mathrm{diag}(\boldsymbol{D})^{-1}\boldsymbol{A} - \frac{1}{np}\mathrm{diag}(\boldsymbol{D})\mathrm{diag}(\boldsymbol{D})^{-1}\boldsymbol{A} \right\|_2 \\
&= \left\| \left( \mathbf{I} - \frac{\mathrm{diag}(\boldsymbol{D})}{np} \right) \mathrm{diag}(\boldsymbol{D})^{-1}\boldsymbol{A} \right\|_2 \\
&\leq \left\| \mathbf{I} - \frac{\mathrm{diag}(\boldsymbol{D})}{np} \right\|_2 \| \boldsymbol{T} \|_2 \\
&\leq \max_{i\in[n]} \left| 1 - \frac{D_i}{np} \right| \| \boldsymbol{T} \|_2 ,
\end{aligned}$$

where the random value $D_i$ is the $i$-th element of $\boldsymbol{D}$.

We bound $\max\limits_{i\in[n]} \left|1 - \frac{D_i}{np}\right|$. Since we consider the ER model, we have

$$\begin{aligned}
\left| 1 - \frac{D_i}{np} \right| &= \left| \frac{D_i}{n} - p \right| \\
&= \left| \frac{D_i}{n} - \mathbb{E}\left[\frac{D_i}{n}\right] \right| .
\end{aligned}$$

Let $t > 0$. By Hoeffding's inequality (Equation 2.4), for each $i \in [n]$, we have

$$\Pr\left( \left| \frac{D_i}{n} - \mathbb{E}\left[\frac{D_i}{n}\right] \right| \geq t \right) \leq 2e^{-2nt^2},$$

which is a bound for $\left|1 - \frac{D_i}{np}\right|$. However, we aim for bounding $\max\limits_{i\in[n]} \left|1 - \frac{D_i}{np}\right|$. By the addition law of probability (Equation 2.2), we have

$$\begin{aligned}
\Pr\left( \exists i \in [n], \left| \frac{D_i}{n} - \mathbb{E}\left[\frac{D_i}{n}\right] \right| \geq t \right) &\leq \sum_{i\in[n]} \Pr\left( \left| \frac{D_i}{n} - \mathbb{E}\left[\frac{D_i}{n}\right] \right| \geq t \right) \\
&\leq 2ne^{-2nt^2}.
\end{aligned}$$

Fixing $0 < \delta \ll 1$ and $t = \sqrt{\frac{\log(2n/\delta)}{2n}}$, we get

$$\begin{aligned}
\Pr\left( \exists i \in [n], \left| \frac{D_i}{n} - \mathbb{E}\left[\frac{D_i}{n}\right] \right| \geq \sqrt{\frac{\log(2n/\delta)}{2n}} \right) &\leq 2ne^{-\log(2n/\delta)} \\
&= 2ne^{\log(\delta/2n)} \\
&= \delta.
\end{aligned}$$

Therefore, with probability $1 - \delta$,

$$\max_{i\in[n]} \left| 1 - \frac{D_i}{np} \right| < \sqrt{\frac{\log(2n/\delta)}{2n}},$$

which, in terms of $n$, is similar to a result in the article by G. Chung and Radcliffe [GR11].

We proceed by the expression of $\|\boldsymbol{T}\|_2$. As discussed earlier, the largest singular value of an observation of $\boldsymbol{T}$ is 1. Thus, by Equation 2.1, we have $\|\boldsymbol{T}\|_2 = 1$.

Hence, with probability $1-\delta$,

$$\begin{aligned}\left\|\boldsymbol{T} - \frac{1}{np}\boldsymbol{A}\right\|_2 &\leq \max_{i\in[n]}\left|1 - \frac{D_i}{np}\right| \|\boldsymbol{T}\|_2 \\ &< \sqrt{\frac{\log(2n/\delta)}{2n}}. \end{aligned} \tag{3.3}$$

The third component $\left\|\frac{1}{np}\boldsymbol{A} - \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n}\right\|_2$. Oliveira [Oli09] shows that for an ER random graph, with probability $1-\delta$,

$$\begin{aligned}\left\|\boldsymbol{A} - p\mathbf{1}\mathbf{1}^{\mathrm{T}}\right\|_2 &\leq 4\sqrt{(n-1)p\log(n/\delta)} \\ &< 4\sqrt{np\log(n/\delta)}.\end{aligned}$$

We rearrange the previous result, and thus arrive to the expression where, with probability $1-\delta$, we have the following:

$$\left\|\frac{1}{np}\boldsymbol{A} - \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n}\right\|_2 < 4\sqrt{\frac{\log(n/\delta)}{np}}. \tag{3.4}$$

Composition. We recompose the three components (Equation 3.2, Equation 3.3, and Equation 3.4). Hence, with high probability,

$$\begin{aligned}\lim_{\mathrm{t_{go}}\to\infty}\left\|\boldsymbol{T}^{\mathrm{t_{go}}} - \frac{\mathbf{1}^{\mathrm{T}}\mathbf{1}}{n}\right\|_2 &< \mathcal{O}\left(\frac{\log n}{\sqrt{n}}\right) + \sqrt{\frac{\log(2n/\delta)}{2n}} + 4\sqrt{\frac{\log(n/\delta)}{np}} \\ &= \mathcal{O}\left(\frac{\log n}{\sqrt{n}}\right),\end{aligned}$$

when $p \in \left[\frac{(\log n)^2}{n}, 1 - \frac{(\log n)^2}{n}\right]$. □

By Theorem 1, we conclude that the spectral norm of the difference between the linear transformation due to the simple gossip algorithm (Algorithm 1) upon convergence and the linear transformation $\frac{\mathbf{1}^{\mathrm{T}}\mathbf{1}}{n}$ is $\mathcal{O}\left(\frac{\log n}{\sqrt{n}}\right)$, when the simple gossip algorithm is executed on a graph modeled by the ER model. Also, we remark that the aforementioned spectral norm is an asymptotic guarantee for the error between and the true average of individual values computed by the simple gossip algorithm (Algorithm 1) and the true average of individual values, when the graph of agents is non-regular yet close to a regular graph (as the expected degree of each vertex of a random graph that follows the ER model is the same).

## 3.3 Averaging on arbitrary graphs

In this section, we continue to study the bias between the average of individual values computed by the simple gossip algorithm (Algorithm 1) and the true average of individual values, when the graph $G = (V, E)$ of agents is non-regular, as initially discussed in Subsection 3.1.2 and Section 3.2.

In particular, we prove an asymptotic guarantee for the mean squared error between the average of sensitive attributes and the average of locally privatized attributes computed by a bias-correcting variant of the simple gossip algorithm (Algorithm 1), when the individual values depend on the degrees of the agents, and the degrees are sensitive attributes. Unlike in Section 3.2 where the graph of agents is modeled by the ER model, we consider the case where the graph of agents is modeled by the configuration model (Definition 4). This way, we consider graphs with arbitrary degree sequences. Also, we consider handshake-free interaction (discussed in Section 2.3).

### 3.3.1 Introduction

In existing works on distributed averaging, it is common to assume a form of handshakes between every two neighboring agents in every exchange of information, i.e., when one agent uses information of a second agent, the second agent becomes aware of this and must actively help the process. However, there exist practical scenarios where such interaction is time consuming due to inactivity of some agents.

To give an illustrative example on handshake-free interaction, let us consider a group of researchers who aim at solving a particular problem. The researchers can follow each other, and thus read each other's currently best strategy in each other's most recently published paper. The researchers work on their solution strategy individually and without necessarily directly contacting each other, attempting to improve their current solution strategies based on their individual skills and the ideas read in the papers of the followed colleagues. At some point, some of them might find a fully satisfactory solution.

We proceed by elaborating on our setting. We model the network of agents by a graph where neighboring agents are connected by edges. Each agent has sensitive attributes, such as its degree, and each agent intends to privatize (add appropriate noise to) its sensitive attributes on its own (without a central curator) before sharing them with other agents. Otherwise, if the sensitive attributes are known, they might be used to profile an agent (e.g., in social media, the connectivity of an agent suggests its popularity).

Our approach for collaboratively computing an unbiased estimate of the average of sensitive attributes has two parts. In the first part, the agents commit to the simple gossip algorithm (Algorithm 1) for distributed averaging. More specifically, each agent hides its sensitive attributes under noise of edge-differential privacy (a standard of data privacy) and makes them visible to the neighbors. Then, each agent repeatedly averages the values revealed by its neighbors and updates the displayed value by the computed average. At some point this converges and the computed averages are biased when the graph is non-regular. In the second part,

the agents perform a procedure combining the biased averages for bias correction. The proposed approach is decentralized, handshake-free, and privacy-preserving.

We investigate a use case for the proposed approach, namely, fitting linear regression models while keeping the degrees private. We prove the asymptotic guarantee that the mean squared error (MSE) between the average of privatized attributes computed by our approach and the average of sensitive attributes is $\mathcal{O}(\frac{1}{n})$, where $n$ is the number of agents. We show on a synthetic graph dataset that the expected error is sufficiently tight. Also, we show on the synthetic graph dataset and real graph datasets that, when the features are polynomials of degrees, the regression model fitted by our approach can outperform the solution when locally privatized attributes are averaged by centralized averaging (`Cen`), i.e., the averaging function $f(\mathbf{x}) = \frac{1}{n}\sum_{i\in[n]} x_i$, where $\mathbf{x} \in \mathbb{R}^n$.

We briefly motivate several elements of our setting. Usually, there are two common forms of information flow [Gia11]: pulling, where an agent asks its neighboring agent for its value, and pushing, where an agent sends its value to a neighboring agent. In this work, we study a weak form of pulling where an agent obtains the current value of a neighbor without the neighbor being aware of this. In particular, each agent continuously publishes its current values so that its neighbors can obtain them, but there is no other communication (e.g., there is no communication process to build overlay networks [JMB09] that can improve distributed computations). We argue that pulling results in a stronger self-management because an agent does not have to wait until another agent disseminates its current values. Since the agents do not exchange handshakes (like in the Transport Layer Security protocol), information dissemination is more robust upon inactive agents. Further, we assume that the degree is sensitive because it can reveal information about the profile of an agent [Hay+09; Hay+10]. Finally, we mention that our approach applies for graphs with power-law degree sequences which, as indicated by Zipf's law, are common in real-world (e.g., computer, social, biological) networks.

**Outline.** In Subsection 3.3.2, we precise our setting. In Subsection 3.3.3, we perform the literature study. In Subsection 3.3.4, we state our approach. In Subsection 3.3.5, we discuss a use case on linear regression. In Subsection 3.3.6, we prove the asymptotic guarantee that the MSE between the average of privatized attributes computed by our approach and the average of sensitive attributes is $\mathcal{O}(\frac{1}{n})$. In Subsection 3.3.7, we discuss the experiments on a synthetic graph dataset and real graph datasets. In Subsection 3.3.8, we conclude.

### 3.3.2 Preliminaries

In this subsection, we precise our setting by describing the communication model and the threat model.

**Communication model.** We model our network of agents by an undirected graph $G = (V, E)$, where $V$ is the set of vertices and $E$ is the set of edges. We denote the order of the graph by $n$. We denote the degree of a vertex $v$ by $d_v$. We denote the degree sequence of $G$ by $\mathbf{d} = (d_1, d_2, \ldots, d_n)$. We denote the lowest degree and the highest degree, respectively, by $d_{\min}$ and $d_{\max}$. For $k \in \mathbb{R}$ and

$\mathbf{x} \in \mathbb{R}^n$, we denote the $k$-th raw moment $\frac{1}{n}\sum_{i\in[n]} x_i^k$ by $\mu_{x^k}$. Similarly, for $k, l \in \mathbb{R}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ , we denote $\frac{1}{n}\sum_{i\in[n]} x_i^k y_i^l$ by $\mu_{x^k.y^l}$. The summaries of notation are provided in Table 2.1 and Table 2.3.

We model $G$ by an observation of a random graph (with fixed vertices and drawn edges) drawn from the configuration model (Definition 4) because the configuration model is parametrized by an arbitrary degree sequence.

We assume that $G$ is connected and has a self-loop or a cycle of odd length because those conditions are necessary for Algorithm 1 to converge. We remark that the connectivity of the configuration model is discussed in Federico and van der Hofstad [FH17]. Then, we remark that a triangle graph is a cycle of odd length, and the count of triangle subgraphs in the configuration model is discussed in Gao et al. [Gao+20]

**Threat model.** We assume that the agents are honest-but-curious [Gol04], i.e., all agents follow the established protocols (they are honest), but they try to use the available information to infer sensitive information of other agents (they are curious). We remark that in our setting where agents publish information and can see published information from others but no other communication is possible, it is straightforward to protect against agents which are malicious in the sense they deviate from the protocol in order to obtain more information. Protecting against agents which deviate from the protocol to influence the result of the computation, e.g., by publishing false information, which is also called data poisoning and studied in Sabater et al. [SBR22], is outside the scope of this work.

We highlight that we use $(\epsilon, \delta)$-edge-differential privacy (we recall its definition below). In particular, we use local differential privacy [Kas+11], where privatization noise is added to attributes; we refer to such attributes as sensitive attributes.

Since we consider datasets of graphs, we remark that classic notions of differential privacy for graphs are edge-differential privacy and node-differential privacy [Kas+13; Hay+09]. In node-differential privacy, adjacent datasets of graphs differ in one vertex and all edges incident with that vertex. In edge-differential privacy, adjacent datasets of graphs differ in one edge. This way, node-differential privacy is, typically, considered as a stronger notion of privacy than edge-differential privacy [TEN22]. However, as indicated in Wang and Wu [WW13], node-differential privacy commonly leads to privatization noise that strongly perturbs privatized attributes. We remark that $k$-edge-differential privacy [Hay+09] is a notion in between edge-differential privacy and node-differential privacy, and under such notion adjacent datasets of graphs differ in $k$ edges.

In this work, we chose to consider edge-differential privacy as opposed to $k$-edge-differential privacy (or node differential privacy) due to the following reasons:

- We consider honest-but-curious agents as our threat model and exclude the threat of collusion among the neighborhood of a particular agent. If the collusion was taken into account, colluding agents would know all the edges of the aforementioned agent and thus would be able to compute its sensitive degree. In our case, each agent knows only one edge of its neighbor. Thus, we consider adjacent datasets of graphs which differ in one edge as opposed to $k$ edges, where such $k$ edges would be known to a set of $k$ colluding agents.

- We consider handshake-free interaction among agents. In such case, an adversary could infer additional information about the size of the neighborhood of a particular agent due to a delay caused to perform a handshake. Thus, we consider adjacent dataset of graphs which differ in one edge as opposed to $k$ edges, where information about such $k$ edges could be inferred if additional $k$ agents sent requests to the agent that is in the process of performing a handshake with the adversary.

- We model the graph of agents by the configuration model (Definition 4) which is parametrized by an arbitrary degree sequence. However, we are mostly interested in graphs with power-law degree sequences, and power-law degree sequences result, typically, in a relatively low average degree of the graph (in our experiments, we will either consider graphs with power-law degree sequences or graphs with a relatively low average degree). This way, we mostly consider cases where the probability of having many subsets of common neighbors is relatively low (i.e., the probability of having many short cycles in a graph is relatively low). As a consequence, it is relatively unlikely that an agent can correctly guess that its two neighbors are also neighbors among themselves.

We recall the definition of edge-differential privacy for graphs by stating it in a similar form as for $(\epsilon, \delta)$-differential privacy in Definition 5.

**Definition 8.** *Let $\epsilon, \delta \geq 0$. A randomized algorithm $\mathcal{A}$ is $(\epsilon, \delta)$-edge-differentially private if and only if, for all $\mathcal{S} \subseteq \text{image}(\mathcal{A})$ and for all triples $(\mathcal{D}, \mathcal{D}', e)$ in a collection where adjacent datasets $\mathcal{D}, \mathcal{D}'$ differ in an edge $e$, we have*

$$\Pr(\mathcal{A}(\mathcal{D}) \in \mathcal{S}) \leq \exp(\epsilon) \Pr(\mathcal{A}(\mathcal{D}') \in \mathcal{S}) + \delta.$$

A classic strategy to guarantee $(\epsilon, \delta)$-differential privacy is to generate noisy values from the Gaussian mechanism (Definition 6). Let $G = (V, E)$ and $G' = (V, E')$ be graphs of adjacent datasets, where $e \in E$ and $e \notin E'$. For each $v \in V$, let $E_v$ and $E'_v$ denote, respectively, the set of edges incident with $v$ on $G$ and $G'$. Let $f$ be a map from a set of all edges incident with a vertex $v$ to its degree, i.e., $d_v = |E_v| = f(E_v)$ and $d'_v = |E'_v| = f(E'_v)$. Based on the global sensitivity discussed in Imola et al. [IMC21], we recall the $l_2$ sensitivity of the degree of an agent under local $(\epsilon, \delta)$-edge-differential privacy:

$$\begin{aligned}
\Delta_2(f) &= \max_{v \in V} \|f(E_v) - f(E'_v)\|_2 \\
&= \max_{v \in V} |d_v - d'_v| \\
&= |d_v - d_v - 1| \\
&= 1.
\end{aligned}$$

### 3.3.3 Literature study

In this subsection, we perform the literature study.

We highlight that we intend to design a communication protocol for distributed averaging, where (i) individual values that depend on the degrees are differentially private, (ii) the averages of individual values are unbiased, (iii) the choice of the degree sequence is arbitrary, (iv) handshakes are absent, and (v) a central curator is absent.

We proceed by discussing some reference works with partial solutions.

If the degree was not a sensitive attribute, our problem could be solved by a gossip algorithm that corrects the bias by an application of the Metropolis–Hastings algorithm [Has70] (as discussed in Subsection 3.1.2). More specifically, in each gossip iteration, an agent could make use of the degrees of its neighbors to correct the bias from each value that is averaged.

If handshakes among agents were allowed, the community could solve the problem by agreeing on an overlay network (a network "built" on top of the initial one). For example, the community could agree on an overlay network that is a regular graph [Hua+21] (as discussed in Subsection 3.1.1).

There are several gossip algorithms that solve our problem by relying on handshakes or pushing. Boyd et al. [Boy+06] requires the agents to agree on an independent edge set (so-called matching). Kempe et al. [KDG03] assumes that each agent knows if a sent message failed to reach its destination. Bellet et al. [BGH20] assumes that an agent always accepts a message sent to it. In Dellenbach et al. [DBR18], every two neighbors initialize their communication by sharing a value related to a positive privatization noise for one and a negative privatization noise for other. Ridgley et al. [RFL19] explicitly mentions the use of pushing.

We remark that garbled circuits [Gas+17; Nik+13] is an alternative approach based on secure multi-party computation. However, secure multi-party computation results in additional computational cost and involves public-key cryptography which relies on handshakes.

We discuss the reference works that we took as building blocks. Oliveira [Oli09] proves an asymptotic guarantee on the matrix norm between the adjacency matrix of an Erdős–Rényi random graph and its expectation, such guarantee has motivated us to work on a similar idea for graphs with arbitrary degree sequences. Iutzeler et al. [ICH13] discusses a gossip algorithm for distributed averaging on graphs with arbitrary degree sequences. Chierichetti et al. [CLP11] considers graph models characterized by power-law degree sequences. Lindell and Pinkas [LP09] provides a compendium on privacy-preserving distributed averaging techniques. Aysal et al. [Ays+09] considers a gossip algorithm that is asynchronous. Bell et al. [Bel+20] fits a regression model using unbiased averages which are computed in a distributed manner.

### 3.3.4 Approach

In this subsection, we firstly briefly discuss the simple gossip algorithm (Algorithm 1), where in later references the simple gossip algorithm is shorthanded by `SiGo`. Then, we state a theorem illustrating that `SiGo` results in biased averages when the graph $G$ of agents is non-regular. Afterwards, we design a strategy for bias correction.

In `SiGo` (Algorithm 1), the transition matrix $\mathbf{T}$ is defined using the adjacency matrix $\mathbf{A}$, which indicates that the averaging by `SiGo` is synchronous. However, Boyd et al. [Boy+06] indicates that, upon convergence, `SiGo` provides the same output when executed asynchronously. Regarding the convergence of `SiGo` (the elements of the output vector $\mathbf{z}$ become equal), Kermarrec and van Steen [KS07] indicates that, in general, $\mathrm{t_{go}} = \lceil \log n \rceil$ is sufficient when $\log n$ is approximately the diameter of $G$ and when the degree sequence $\mathbf{d}$ follows a power-law distribution.

We state a theorem for the value of each element of the output $\mathbf{z}$ of `SiGo`, when `SiGo` converges:

**Theorem 2.** *Let $\mathbf{A}$ be the adjacency matrix of a graph $G$ of order $n$. Let $\mathbf{d}$ be the degree sequence of $G$. Let $\mathbf{w} \in \mathbb{R}^n$. Let $\texttt{SiGo}\,(\mathbf{A}, \mathrm{t_{go}}, \mathbf{w})$ denote any element of the output $\mathbf{z}$ of Algorithm 1. We have*

$$\lim_{\mathrm{t_{go}} \to \infty} \texttt{SiGo}\,(\mathbf{A}, \mathrm{t_{go}}, \mathbf{w}) = \frac{1}{n} \frac{1}{\mu_d} \sum_{i \in [n]} d_i w_i, \tag{3.5}$$

*when $G$ is connected and has a self-loop or a cycle of odd length.*

In later references, $\texttt{SiGo}\,(\mathbf{w})$ shorthands $\lim_{\mathrm{t_{go}} \to \infty} \texttt{SiGo}\,(\mathbf{A}, \mathrm{t_{go}}, \mathbf{w})$.

*Proof.* In this proof, we expand $\lim_{\mathrm{t_{go}} \to \infty} \mathbf{T}^{\mathrm{t_{go}}} \mathbf{w}$, where

$$\mathbf{T} = \mathrm{diag}(\mathbf{d})^{-1} \mathbf{A}$$

is the transition matrix of `SiGo` (Algorithm 1), and where $\mathbf{w} \in \mathbb{R}^n$ contains the values that are intended to be averaged.

Even though $\mathbf{T}$ is not symmetric (it is a row stochastic matrix), we can define the matrix

$$\mathbf{X} = \mathrm{diag}(\mathbf{d})^{1/2} \mathbf{T} \mathrm{diag}(\mathbf{d})^{-1/2},$$

similarly as in the article by G. Chung and Radcliffe [GR11]. This way, $\mathbf{X}$ is symmetric.

Both $\mathbf{T}$ and $\mathbf{X}$ have a largest singular value 1, which has multiplicity 1 if $G$ is connected, and this suggests that

$$\mathbf{T}\mathbf{1} = \mathbf{1}$$

is an eigenvector of $\mathbf{T}$, which is paired with the eigenvalue 1. Similarly,

$$\mathbf{X}\mathrm{diag}(\mathbf{d})^{1/2}\mathbf{1} = \mathrm{diag}(\mathbf{d})^{1/2}\mathbf{1}$$

is an eigenvector of $\mathbf{X}$, which is paired with the eigenvalue 1. We normalize the aforementioned eigenvector so that its elements sum to 1:

$$\mathbf{v}_1 = \frac{\mathrm{diag}(\mathbf{d})^{1/2}\mathbf{1}}{\sqrt{n\mu_d}}.$$

Since $\mathbf{X}$ is symmetric, it has real eigenvalues and orthogonal eigenvectors. Let $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$ be the eigendecomposition of $\mathbf{X}$, where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues in decreasing order, implying $\mathbf{\Lambda}_{1,1} = 1$. This way, we have $\mathbf{U}_{:,1} = \mathbf{v}_1$.

Let $\mathbf{I}_n$ denote the $n \times n$ identity matrix. We derive the limit when the elements of $\mathbf{w}$ are iteratively averaged using $\mathbf{T}$:

$$\begin{aligned}
\lim_{\mathrm{t_{go}}\to\infty} \mathbf{T}^{\mathrm{t_{go}}}\mathbf{w} &= \lim_{\mathrm{t_{go}}\to\infty} \left(\mathrm{diag}(\mathbf{d})^{-1/2}\mathbf{X}\mathrm{diag}(\mathbf{d})^{1/2}\right)^{\mathrm{t_{go}}} \mathbf{w} \\
&= \lim_{\mathrm{t_{go}}\to\infty} \mathrm{diag}(\mathbf{d})^{-1/2}\mathbf{X}^{\mathrm{t_{go}}}\mathrm{diag}(\mathbf{d})^{1/2}\mathbf{w} \\
&= \lim_{\mathrm{t_{go}}\to\infty} \mathrm{diag}(\mathbf{d})^{-1/2}\mathbf{U}\mathbf{\Lambda}^{\mathrm{t_{go}}}\mathbf{U}^{\mathrm{T}}\mathrm{diag}(\mathbf{d})^{1/2}\mathbf{w} \\
&= \mathrm{diag}(\mathbf{d})^{-1/2}\mathbf{U}\mathrm{diag}(1,0,\ldots,0)\mathbf{U}^{\mathrm{T}}\mathrm{diag}(\mathbf{d})^{1/2}\mathbf{w} \\
&= \mathrm{diag}(\mathbf{d})^{-1/2}\mathbf{v}_1\mathbf{v}_1^{\mathrm{T}}\mathrm{diag}(\mathbf{d})^{1/2}\mathbf{w} \\
&= \mathrm{diag}(\mathbf{d})^{-1/2}\frac{\mathrm{diag}(\mathbf{d})^{1/2}\mathbf{1}}{\sqrt{n\mu_d}}\frac{\mathbf{1}^{\mathrm{T}}\mathrm{diag}(\mathbf{d})^{1/2}}{\sqrt{n\mu_d}}\mathrm{diag}(\mathbf{d})^{1/2}\mathbf{w} \\
&= \frac{1}{n\mu_d}\mathbf{I}_n\mathbf{1}\mathbf{1}^{\mathrm{T}}\mathrm{diag}(\mathbf{d})\mathbf{w} \\
&= \frac{1}{n\mu_d}\mathbf{1}\mathbf{1}^{\mathrm{T}}\mathrm{diag}(\mathbf{d})\mathbf{w} \\
&= \mathbf{1}\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i w_i,
\end{aligned}$$

where $\lim_{\mathrm{t_{go}}\to\infty} \Lambda^{\mathrm{t_{go}}}$ exists when the graph is connected and has a self-loop or a cycle of odd length, as indicated by Paz [Paz63], and the second largest singular value of $\Lambda$ is lower than 1, as indicated by Banerjee and Mehatari [BM16].

Hence, when `SiGo` converges, every element of the output $\mathbf{z}$ of `SiGo` (Algorithm 1) is

$$\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i w_i.$$

□

Theorem 2 indicates that the resulting average is biased by $\mu_d$ and $d_i$ (for each $i \in [n]$), when $d_i \neq \mu_d$ (i.e., $G$ is non-regular). That is, the squared difference between any element of the output of $\texttt{SiGo}(\mathbf{w})$ and $\mu_w = \frac{1}{n}\sum_{i\in[n]} w_i$ is non-zero:

$$\begin{aligned}
\left(\texttt{SiGo}(\mathbf{w}) - \mu_w\right)^2 &= \left(\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i w_i - \mu_w\right)^2 \qquad \text{(by Theorem 2)} \\
&= \left(\frac{\mu_{d.w}}{\mu_d} - \mu_w\right)^2. \qquad (3.6)
\end{aligned}$$

**Bias correction.** We provide a procedure to correct the bias illustrated by Equation 3.6. We start by an example where the community computes an unbiased

estimate of $\mu_w$ from two gossiping runs. Firstly, `Sigo` is executed (sequentially or in parallel) with the respective inputs $(w_i d_i^{-1})_{i=1}^n$ and $(d_i^{-1})_{i=1}^n$:

$$\begin{aligned}
\texttt{SiGo}\left((w_i d_i^{-1})_{i=1}^n\right) &= \frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} w_i \qquad \text{(by Theorem 2)} \\
&= \frac{\mu_w}{\mu_d}, \\
\texttt{SiGo}\left((d_i^{-1})_{i=1}^n\right) &= \frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} 1 \\
&= \frac{1}{\mu_d}. \qquad (3.7)
\end{aligned}$$

Then, the community can compute

$$\frac{\texttt{SiGo}\left((w_i d_i^{-1})_{i=1}^n\right)}{\texttt{SiGo}\left((d_i^{-1})_{i=1}^n\right)} = \mu_w. \qquad (3.8)$$

By Theorem 2, we generalize Equation 3.8 and thus define the bias-correcting gossip algorithm (`BCGo`):

$$\begin{aligned}
\frac{\texttt{SiGo}\left((w_i d_i^{-1})_{i=1}^n\right)}{\texttt{SiGo}\left((d_i^{-1})_{i=1}^n\right)} &= \frac{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i(w_i d_i^{-1})}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i(d_i^{-1})} \\
&= \frac{1}{n}\sum_{i\in[n]} w_i \\
&= \mu_w. \qquad (3.9)
\end{aligned}$$

We illustrate an example of applying `BCGo` for computing the order $n$ of a graph $G$, when $G$ is connected and has a cycle of odd length. Similarly as in the example given after recalling `SiGo` (Algorithm 1), we assign a label 1 to one agent, and a label 0 to the remaining $n-1$ agents. When `BCGo` converges, each agent computes the average $\frac{1+0+\ldots+0}{n} = \frac{1}{n}$ (Equation 3.9). By taking the reciprocal of $\frac{1}{n}$, each agent computes $n$.

We highlight that unbiased averages can be generalized by U-statistics (Definition 1). This way, the estimates obtained by `BCGo` (Equation 3.9) are U-statistics with kernel $\phi : x \mapsto x$ of degree 1. U-statistics of degree 1 are present in machine learning applications, for example, in classic strategies for fitting linear regression models, as discussed in Subsection 3.3.5, bootstrap aggregation in random forests, as discussed in Peng et al. [PCM22], and gradient descent in training artificial neural networks, as discussed in McMahan et al. [McM+17]

### 3.3.5 Use case on linear regression

In this subsection, we investigate a use case for the proposed approach, namely, fitting a linear regression model while keeping the degrees private, where each agent is assigned one labeled example, and the features depend on the degrees. Firstly, we define a multiple linear regression model whose features are powers of

the degrees. Then, we show that the regression model can be fitted from averages. Afterwards, we discuss the privatization and the bounding of powers of the degrees. Finally, we provide the pseudo-code for fitting the multiple linear regression model by regularized least squares and when averaging the privatized attributes by `BCGo` (Equation 3.9).

Let $m \in \mathbb{N}$. We define a special case of a multiple linear regression model (defined in Equation 2.8) with $m+1$ regression parameters and a scalar target value. For each $i \in [n]$,

$$y_i = \theta_0 + \theta_1 d_i^{k_1} + \ldots + \theta_m d_i^{k_m} + \xi_i^{\text{reg}}, \tag{3.10}$$

where $\theta_0, \theta_1, \ldots, \theta_m \in \mathbb{R}$ are regression parameters, $d_i^{k_1}, d_i^{k_2}, \ldots, d_i^{k_m} \in \mathbb{R}$ are features ($k_1, k_2, \ldots, k_m \in \mathbb{R} \setminus \{0\}$), $y_i$ is a target value, $\xi_i^{\text{reg}}$ is regression noise which is an independent observation of $\mathcal{N}(0, \sigma_{\text{reg}}^2)$, and $\sigma_{\text{reg}}^2$ is the variance of regression noise.

We remark that, while `BCGo` (Equation 3.9) can compute averages for fitting regression models with arbitrary features, we focus on the multiple linear regression model defined in Equation 3.10 because in this work we limit ourselves to privatization noise for the features that depend on powers of the degrees.

As a consequence, we remark that polynomials of arbitrary degree can approximate other functions (e.g., the logarithm, trigonometric functions, etc.) using the Taylor series (which are discussed in detail in Canuto and Tabacco [CT15]). This way, our regression model can approximate linear regression models with a broad class of features. Regarding the practical applicability of such regression models, we provide a reference to the linear regression datasets that are part of *the NIST Standard Reference Database* (accessible on `itl.nist.gov/div898/strd/lls/lls.shtml`).

Let $\mathbf{X} = \begin{bmatrix} \mathbf{1} & \mathbf{x}_1 & \ldots & \mathbf{x}_m \end{bmatrix} \in \mathbb{R}^{n \times (m+1)}$, where $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ are feature vectors. Let $\mathbf{y}$ be a vector of target values. We fit the regression model defined in Equation 3.10 by regularized least squares (discussed in Subsection 2.4.1). That is, referring to Equation 2.9, we compute the vector $\hat{\theta}$ of parameter estimates as follows:

$$\hat{\theta} = \left( \frac{1}{n} \mathbf{X}^{\mathrm{T}} \mathbf{X} + \lambda \mathbf{I}_{m+1} \right)^{-1} \frac{1}{n} \mathbf{X}^{\mathrm{T}} \mathbf{y}, \tag{3.11}$$

where

$$\begin{aligned} \frac{1}{n} \mathbf{X}^{\mathrm{T}} \mathbf{X} &= \begin{bmatrix} 1 & \frac{1}{n} \sum_{i \in [n]} d_i^{k_1} & \ldots & \frac{1}{n} \sum_{i \in [n]} d_i^{k_m} \\ \frac{1}{n} \sum_{i \in [n]} d_i^{k_1} & \frac{1}{n} \sum_{i \in [n]} d_i^{2k_1} & \ddots & \frac{1}{n} \sum_{i \in [n]} d_i^{k_1} d_i^{k_m} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{1}{n} \sum_{i \in [n]} d_i^{k_m} & \frac{1}{n} \sum_{i \in [n]} d_i^{k_1} d_i^{k_m} & \ldots & \frac{1}{n} \sum_{i \in [n]} d_i^{2k_m} \end{bmatrix} \\ &= \begin{bmatrix} 1 & \mu_{d^{k_1}} & \ldots & \mu_{d^{k_m}} \\ \mu_{d^{k_1}} & \mu_{d^{2k_1}} & \ddots & \mu_{d^{k_1} . d^{k_m}} \\ \vdots & \ddots & \ddots & \vdots \\ \mu_{d^{k_m}} & \mu_{d^{k_1} . d^{k_m}} & \ldots & \mu_{d^{2k_m}} \end{bmatrix}, \end{aligned} \tag{3.12}$$

$$\begin{aligned} \frac{1}{n} \mathbf{X}^{\mathrm{T}} \mathbf{y} &= \begin{bmatrix} \frac{1}{n} \sum_{i \in [n]} y_i & \frac{1}{n} \sum_{i \in [n]} y_i d_i^{k_1} & \ldots & \frac{1}{n} \sum_{i \in [n]} y_i d_i^{k_m} \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} \mu_y & \mu_{y . d^{k_1}} & \ldots & \mu_{y . d^{k_m}} \end{bmatrix}^{\mathrm{T}}, \end{aligned} \tag{3.13}$$

and where $\lambda \in \mathbb{R}$ is the regularization parameter, and $\mathbf{I}_{m+1}$ is the $(m+1) \times (m+1)$ identity matrix.

We highlight that the parameter estimates $\hat{\theta}$ are computed from the averages $\mu_{d^{k_1}}, \mu_{d^{k_2}}, \ldots, \mu_{d^{k_m}}$, $\mu_{d^{2k_1}}, \mu_{d^{2k_2}}, \ldots, \mu_{d^{2k_m}}$, $\mu_{d^{k_1}.d^{k_2}}, \mu_{d^{k_1}.d^{k_3}}, \ldots, \mu_{d^{k_{m-1}}.d^{k_m}}$, $\mu_{y.d^{k_1}}, \mu_{y.d^{k_2}}, \ldots, \mu_{y.d^{k_m}}$, and $\mu_y$ ($2m + \frac{m(m-1)}{2} + m + 1$ of averages in total). By Definition 1, the aforementioned averages are U-statistics of degree 1.

We assume that $d_{\min}$ and $d_{\max}$ are known. For $k \in \mathbb{R} \setminus \{0\}$, we privatize $d_i^k$ using the Gaussian mechanism (Definition 6) as follows:

$$D_{i,k}^{\mathrm{dp}} \sim \mathcal{N}\left(d_i^k, (\sigma_k^{\mathrm{dp}})^2\right), \tag{3.14}$$

$$\sigma_k^{\mathrm{dp}} = \frac{\sqrt{2\log(1.25/\delta')}\Delta_2(d_i^k)}{\epsilon'}, \tag{3.15}$$

where $(\epsilon', \delta')$ is an even split of the total privacy budget $(\epsilon, \delta)$ and, as adjacent datasets of graphs differ in one edge (Definition 8), we have

$$\begin{aligned}\Delta_2(d_i^k) &= \max_{d \in [d_{\min}, d_{\max}-1]} \left|(d+1)^k - d^k\right| \\ &= \begin{cases} d_{\min}^k - (d_{\min}+1)^k & \text{when } k < 0, \\ (d_{\min}+1)^k - d_{\min}^k & \text{when } 0 < k < 1, \\ d_{\max}^k - (d_{\max}-1)^k & \text{when } k \geq 1. \end{cases}\end{aligned} \tag{3.16}$$

We remark that we do not consider the privatization of the power 0 of a degree because it always equals 1 and thus is non-sensitive (it no longer depends on the degree).

**Remark 1.** *A privatized attribute $d_{i,k}^{\mathrm{dp}}$ is an independent observation of $D_{i,k}^{\mathrm{dp}}$ (Equation 3.14). Upon `BCGo`, privatization of $d_i^{k_1-1}, d_i^{k_2-1}, \ldots, d_i^{k_m-1}$ is independent from $d_i^{-1}, d_i$. For clarity, privatized $d_i^{-1}, d_i$ are denoted, respectively, by $d_{i,-1'}^{\mathrm{dp}}, d_{i,1'}^{\mathrm{dp}}$.*

We highlight that we privatize the transformed degree $d_i^k$, as opposed to privatizing $d_i$ and then computing $(d_i^{\mathrm{dp}})^k$. This is done in order to mitigate the risk of $d_i^{\mathrm{dp}}$ being too close to the singularity at the value 0 when $k < 0$. If $d_i^{\mathrm{dp}}$ is close to 0, then the feature $(d_i^{\mathrm{dp}})^k$ would be an outlier disturbing the prediction accuracy of the fitted regression model.

We proceed by discussing the bounds for each privatized attribute $d_{i,k}^{\mathrm{dp}}$ so that it stays centred at $d_i^k$ (preserves unbiasedness). Let

$$a_i = \begin{cases} \min(d_i^k - d_{\max}^k, d_{\min}^k - d_i^k) & \text{when } k < 0, \\ \min(d_i^k - d_{\min}^k, d_{\max}^k - d_i^k) & \text{when } k > 0. \end{cases}$$

This way, we have the following bounds:

$$d_i^k - a_i \leq d_{i,k}^{\mathrm{dp}} \leq d_i^k + a_i. \tag{3.17}$$

**Remark 2.** *If a privatized attribute $d_{i,k}^{\mathrm{dp}}$ is outside the bounds (Equation 3.17), it is resampled uniformly in the interval $[d_i^k - a_i, d_i^k + a_i]$.*

We highlight that, upon `BCGo`, the sensitive attributes of an agent $i$ are $d_i^{k_1-1}, d_i^{k_2-1}, \ldots, d_i^{k_m-1}$, $d_i^{-1}$, $d_i$ ($m+2$ in total). Similarly, upon `Cen`, the sensitive attributes are $d_i^{k_1}, d_i^{k_2}, \ldots, d_i^{k_m}$ ($m$ in total). We leave the following remarks on the privacy guarantees for locally privatizing and then averaging the aforementioned powers of the degrees:

- We assume that $d_{\min}$ and $d_{\max}$ are known. This way, each agent can compute the $l_2$ sensitivity (Equation 3.16) for local privatization using the Gaussian mechanism (Definition 6). We recall that the Gaussian mechanism ensures $(\epsilon, \delta)$-differential privacy.

- We assume that each agent knows only its own edges, and not edges between other vertices. For example, such assumption mitigates the previously discussed threat of collusion among the neighborhood of a particular agent. Thus, the value of the degree of a particular vertex can not be deduced from a set of edges incident to that vertex.

- We assume that the computation of the average of locally privatized attributes is under a sufficiently low precision (i.e., some rounding happens). In other words, the probability of observing any possible (rounded) average should remain sufficiently high. For example, such assumption mitigates the threat of an adversary learning additional information about the topology of the communication network, when the adversary is connected to all the agents and when the adversary observes subsequent averaging. This way, revealed functions (i.e., averages) of locally privatized attributes (i.e., powers of the degrees) remain differentially private.

- Referring to Sei and Ohsuga [SO22], we assume that target values are non-sensitive. Even though a target value $y_i$ (Equation 3.10) is a linear combination of powers of the degree, the exact values of the regression parameters and the regression noise remain unknown, thus the value of the degree can not be deduced.

- Referring to the composition theorem in Dwork and Roth [DR14], privatization of the sensitive attributes upon even splits of the total privacy budget $(\epsilon, \delta)$ lead to $(\epsilon, \delta)$-differential privacy.

- If a privatized attribute is outside the bounds (Equation 3.17), we resample it uniformly, while uniform sampling is $(0, 0)$-differentially private.

In Algorithm 4, we provide the pseudo-code for fitting the multiple linear regression model by regularized least squares (Equation 3.12 and Equation 3.13) and when averaging the privatized attributes by `BCGo` (Equation 3.9).

**Algorithm 4:** `BiasCorrectingGossip` (`BCGo`) for privacy-preserving regression

`/* Privatization (Remark 1) and bounding (Remark 2) */`

$\epsilon', \delta' \leftarrow \frac{\epsilon}{m+2}, \frac{\delta}{m+2}$

**for** $i \in [n]$ **do**

  **for** $j \in [m]$ **do**

    $d^{\mathrm{dp}}_{i,k_j-1} \leftarrow \texttt{privatizeAndBound}(d_i^{k_j-1}, \epsilon', \delta')$

  **for** $k \in \{-1', 1'\}$ **do**

    $d^{\mathrm{dp}}_{i,k} \leftarrow \texttt{privatizeAndBound}(d_i^{k}, \epsilon', \delta')$

`/* Averaging by BCGo (Equation 3.9) */`

$\hat{\mu}_y = \frac{\texttt{SiGo}\left((y_i d^{\mathrm{dp}}_{i,-1})^n_{i=1}\right)}{\texttt{SiGo}\left((d^{\mathrm{dp}}_{i,-1'})^n_{i=1}\right)}$

**for** $j \in [m]$ **do**

  $\hat{\mu}_{d^{k_j}}, \hat{\mu}_{d^{2k_j}}, \hat{\mu}_{y.d^{k_j}} \leftarrow$
  $\frac{\texttt{SiGo}\left((d^{\mathrm{dp}}_{i,k_j-1})^n_{i=1}\right)}{\texttt{SiGo}\left((d^{\mathrm{dp}}_{i,-1'})^n_{i=1}\right)}, \frac{\texttt{SiGo}\left(((d^{\mathrm{dp}}_{i,k_j-1})^2 d^{\mathrm{dp}}_{i,1'})^n_{i=1}\right)}{\texttt{SiGo}\left((d^{\mathrm{dp}}_{i,-1'})^n_{i=1}\right)}, \frac{\texttt{SiGo}\left((y_i d^{\mathrm{dp}}_{i,k_j-1})^n_{i=1}\right)}{\texttt{SiGo}\left((d^{\mathrm{dp}}_{i,-1'})^n_{i=1}\right)}$

  **for** $j' \in \{j, \ldots, m-1\}$ **do**

    $\hat{\mu}_{d^{k_j}.d^{k_{j'}}} = \frac{\texttt{SiGo}\left((d^{\mathrm{dp}}_{i,k_j-1} d^{\mathrm{dp}}_{i,k_{j'}-1} d^{\mathrm{dp}}_{i,1'})^n_{i=1}\right)}{\texttt{SiGo}\left((d^{\mathrm{dp}}_{i,-1'})^n_{i=1}\right)}$

`/* Fitting by regularized least squares (Equation 3.11) */`

$\hat{\theta} \leftarrow \texttt{fit}(\hat{\mu}_y, \hat{\mu}_{d^{k_1}}, \ldots, \hat{\mu}_{d^{2k_1}}, \ldots, \hat{\mu}_{y.d^{k_1}}, \ldots, \hat{\mu}_{d^{k_1}.d^{k_2}}, \ldots, \lambda)$

The pseudo-code in Algorithm 4 consists of the following three parts. In the first part, we privatize and bound powers of the degrees (the corresponding privacy guarantees are discussed in the list above). In the second part, we execute `SiGo` (Algorithm 1) on several functions whose arguments are locally privatized attributes and target values; such revealed functions remain differentially private because they do not reveal sensitive attributes. In the third part, each agent simply performs regularized least squares (Equation 3.11), which does not reveal sensitive attributes neither.

### 3.3.6 Error analysis

In this subsection, we define the expected errors between the average (e.g., $\mu_{d^{k_1}}$ in Equation 3.12) of sensitive attributes and the average of privatized attributes computed, respectively, by `Cen` and `BCGo` (Equation 3.9). Then, we discuss the conditions when the average computed by `BCGo` results in less privatization noise compared to the average computed by `Cen`. Finally, we define the respective empirical errors for evaluating the aforementioned expected errors.

Let $k \in \mathbb{R} \setminus \{0\}$. We define the average ($k$-th raw moment) of sensitive attributes:

$$\begin{aligned} s_k &= \mu_{d^k} \\ &= \frac{1}{n} \sum_{i \in [n]} d_i^k. \end{aligned} \tag{3.18}$$

We define the averages of privatized attributes (Equation 3.14) computed, respectively, by `Cen` and `BCGo`:

$$S_k^{\texttt{Cen}} = \frac{1}{n}\sum_{i\in[n]} D_{i,k}^{\mathrm{dp}}, \tag{3.19}$$

$$S_k^{\texttt{BCGo}} = \frac{\texttt{SiGo}\left((D_{i,k-1}^{\mathrm{dp}})_{i=1}^n\right)}{\texttt{SiGo}\left((D_{i,-1}^{\mathrm{dp}})_{i=1}^n\right)}. \tag{3.20}$$

We highlight that our error analysis is limited to sensitive attributes that are powers (excluding the power 0) of the degrees. Otherwise, the error analysis would be more complex because the numerator of Equation 3.9 would include the multiplication between the following two terms with privatization noise: the privatized $w_i$ and the privatized $d_i^{-1}$.

**Expected errors.** We define the expected error between the average (Equation 3.18) of sensitive attributes and the average (Equation 3.19) of privatized attributes computed by `Cen`:

$$e_k^{\texttt{Cen}} = \mathbb{E}\left[\left(s_k - S_k^{\texttt{Cen}}\right)^2\right], \tag{3.21}$$

We state a theorem (a folklore result on the expected value of the squared average of Gaussian random variables) quantifying the expected error $e_k^{\texttt{Cen}}$ (Equation 3.21).

**Theorem 3.** *We have*

$$e_k^{\texttt{Cen}} = \frac{1}{n}(\sigma_k^{\mathrm{dp}})^2, \tag{3.22}$$

*where $\sigma_k^{\mathrm{dp}}$ is given in Equation 3.15. This way, $e_k^{\texttt{Cen}} = \mathcal{O}\left(\frac{1}{n}\right)$.*

*Proof.* We intend to expand $e_k^{\texttt{Cen}}$. We start as follows:

$$\begin{aligned} e_k^{\texttt{Cen}} &= \mathbb{E}\left[\left(s_k - S_k^{\texttt{Cen}}\right)^2\right] && \text{(by Eqn 3.21)} \\ &= \mathbb{E}\left[\left(\mu_{d^k} - S_k^{\texttt{Cen}}\right)^2\right] && \text{(by Eqn 3.18)} \\ &= \mu_{d^k}^2 - 2\mu_{d^k}\mathbb{E}\left[S_k^{\texttt{Cen}}\right] + \mathbb{E}\left[\left(S_k^{\texttt{Cen}}\right)^2\right]. && (3.23) \end{aligned}$$

Then, we derive

$$\begin{aligned} \mathbb{E}[S_k^{\texttt{Cen}}] &= \mathbb{E}\left[\frac{1}{n}\sum_{i\in[n]} D_{i,k}^{\mathrm{dp}}\right] && \text{(by Eqn 3.19)} \\ &= \frac{1}{n}\sum_{i\in[n]} \mathbb{E}\left[D_{i,k}^{\mathrm{dp}}\right] \\ &= \frac{1}{n}\sum_{i\in[n]} d_i^k && \text{(by Eqn 3.14)} \\ &= \mu_{d^k}, && (3.24) \end{aligned}$$

$$\begin{aligned}
\operatorname{var}\left(S_k^{\texttt{Cen}}\right) &= \operatorname{var}\left(\frac{1}{n}\sum_{i\in[n]} D_{i,k}^{\mathrm{dp}}\right) && \text{(by Eqn 3.19)}\\
&= \frac{1}{n^2}\operatorname{var}\left(\sum_{i\in[n]} D_{i,k}^{\mathrm{dp}}\right)\\
&= \frac{1}{n^2}\sum_{i\in[n]} \operatorname{var}\left(D_{i,k}^{\mathrm{dp}}\right)\\
&= \frac{1}{n}(\sigma_k^{\mathrm{dp}})^2. && \text{(by Eqn 3.15)} \qquad (3.25)
\end{aligned}$$

Further, for a random variable $Z$, we state an identity:

$$\mathbb{E}[Z^2] = \operatorname{var}(Z) + \mathbb{E}^2[Z]. \qquad (3.26)$$

We continue the derivation in Equation 3.23:

$$\begin{aligned}
e_k^{\texttt{Cen}} &= \mu_{d^k}^2 - 2\mu_{d^k}\mathbb{E}\left[S_k^{\texttt{Cen}}\right] + \operatorname{var}\left(S_k^{\texttt{Cen}}\right) + \mathbb{E}^2[S_k^{\texttt{Cen}}] && \text{(by Eqn 3.26)}\\
&= \mu_{d^k}^2 - 2\mu_{d^k}^2 + \frac{1}{n}(\sigma_k^{\mathrm{dp}})^2 + \mu_{d^k}^2 && \text{(by Eqn 3.25, Eqn 3.24)}\\
&= \frac{1}{n}(\sigma_k^{\mathrm{dp}})^2. && (3.27)
\end{aligned}$$

Hence, we conclude that $e_k^{\texttt{Cen}} = \mathcal{O}(\frac{1}{n})$. □

We define the expected error between the average (Equation 3.18) of sensitive attributes and the average (Equation 3.20) of privatized attributes computed by `BCGo`:

$$e_k^{\texttt{BCGo}} = \mathbb{E}\left[\left(s_k - S_k^{\texttt{BCGo}}\right)^2\right]. \qquad (3.28)$$

We provide a theorem quantifying the expected error $e_k^{\texttt{BCGo}}$ (Equation 3.28) for the non-asymptotic case.

**Theorem 4.** *Let* $\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{-1}^{\mathrm{dp}})^2 \neq 0$ *and* $\frac{1}{\mu_d} - 3\frac{1}{\sqrt{n}}\frac{\sqrt{\mu_{d^2}}}{\mu_d}\sigma_{-1}^{\mathrm{dp}} > 0$, *where* $\mu_{d^k}$ *and* $\sigma_k^{\mathrm{dp}}$ *are given, respectively, in Equation 3.18 and Equation 3.15. With high probability,*

$$\begin{aligned}
e_k^{\texttt{BCGo}} \approx \mu_{d^k}^2 - 2\frac{\mu_{d^k}^2}{\mu_d}\sqrt{n}\frac{\sqrt{2}\mu_d}{\sqrt{\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}} f_{\mathrm{daw}}\left(\frac{\sqrt{n}}{\sqrt{2\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}}\right) + \left(\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{k-1}^{\mathrm{dp}})^2 + \frac{\mu_{d^k}^2}{\mu_d^2}\right)\\
n\frac{\mu_d^2}{\mu_{d^2}(\sigma_{-1}^{\mathrm{dp}})^2}\left(\sqrt{n}\frac{\sqrt{2}}{\sqrt{\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}} f_{\mathrm{daw}}\left(\frac{\sqrt{n}}{\sqrt{2\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}}\right) - 1\right), \qquad (3.29)
\end{aligned}$$

*where* $f_{\mathrm{daw}}(x) = e^{-x^2}\int_0^x e^{t^2}\,dt$. *(The approximation is due to the absence of closed-form expressions for the mean and the variance of the reciprocal of a Gaussian random variable, and the quantification of the approximation error is outside the scope of this work.)*

*Proof.* We intend to expand $e_k^{\texttt{BCGo}}$. We start as follows:

$$\begin{aligned}
e_k^{\texttt{BCGo}} &= \mathbb{E}\left[\left(s_k - S_k^{\texttt{BCGo}}\right)^2\right] \quad \text{(by Eqn 3.28)} \\
&= s_k^2 + \mathbb{E}\left[\left(S_k^{\texttt{BCGo}}\right)^2\right] - 2s_k\mathbb{E}\left[S_k^{\texttt{BCGo}}\right] \\
&= s_k^2 + \mathbb{E}\left[\left(\texttt{SiGo}\left((D_{i,k-1}^{\mathrm{dp}})_{i=1}^n\right)\right)^2\right]\mathbb{E}\left[\left(\frac{1}{\texttt{SiGo}\left((D_{i,-1}^{\mathrm{dp}})_{i=1}^n\right)}\right)^2\right] \\
&\quad - 2s_k\mathbb{E}\left[\texttt{SiGo}\left((D_{i,k-1}^{\mathrm{dp}})_{i=1}^n\right)\right]\mathbb{E}\left[\frac{1}{\texttt{SiGo}\left((D_{i,-1}^{\mathrm{dp}})_{i=1}^n\right)}\right] \quad \text{(by Eqn 3.20)} \\
&= s_k^2 + \mathbb{E}\left[\left(\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,k-1}^{\mathrm{dp}}\right)^2\right]\mathbb{E}\left[\left(\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right)^2\right] \quad \text{(by Thm 2)} \\
&\quad - 2s_k\mathbb{E}\left[\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,k-1}^{\mathrm{dp}}\right]\mathbb{E}\left[\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right]. \qquad (3.30)
\end{aligned}$$

Then, we express

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,k-1}^{\mathrm{dp}}\right] &= \frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i\mathbb{E}\left[D_{i,k-1}^{\mathrm{dp}}\right] \\
&= \frac{\mu_{d^k}}{\mu_d}, \quad \text{(by Eqn 3.14)} \qquad (3.31)
\end{aligned}$$

$$\begin{aligned}
\mathrm{var}\left(\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,k-1}^{\mathrm{dp}}\right) &= \frac{1}{n^2}\frac{1}{\mu_d^2}\sum_{i\in[n]} d_i^2\mathrm{var}\left(D_{i,k-1}^{\mathrm{dp}}\right) \\
&= \frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{k-1}^{\mathrm{dp}})^2. \quad \text{(by Eqn 3.15)} \qquad (3.32)
\end{aligned}$$

Thus,

$$\begin{aligned}
\mathbb{E}\left[\left(\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,k-1}^{\mathrm{dp}}\right)^2\right] &= \mathrm{var}\left(\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,k-1}^{\mathrm{dp}}\right) + \mathbb{E}^2\left[\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,k-1}^{\mathrm{dp}}\right] \\
&\quad \text{(by Eqn 3.26)} \\
&= \frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{k-1}^{\mathrm{dp}})^2 + \frac{\mu_{d^k}^2}{\mu_d^2}. \quad \text{(by Eqn 3.32, Eqn 3.31)} \qquad (3.33)
\end{aligned}$$

We use two heuristics for a Gaussian random variable $Z$ with mean $\mu$ and variance $\sigma^2$. That is, when $\mu \neq 0$, $\sigma \neq 0$, and $|\mu| - 3\sigma > 0$, with high probability,

$$\mathbb{E}\left[\frac{1}{Z}\right] \approx \frac{\sqrt{2}}{\sigma} f_{\mathrm{daw}}\left(\frac{\mu}{\sqrt{2}\sigma}\right), \qquad (3.34)$$

$$\mathbb{E}\left[\frac{1}{Z^2}\right] \approx \frac{1}{\sigma^2}\left(\mu\frac{\sqrt{2}}{\sigma} f_{\mathrm{daw}}\left(\frac{\mu}{\sqrt{2}\sigma}\right) - 1\right), \tag{3.35}$$

where $f_{\mathrm{daw}}(x) = e^{-x^2}\int_0^x e^{t^2}\,dt$ is the Dawson function. Peng [Pen08] shows that Equation 3.34 holds in the Cauchy principal value. When $\mu = 0.9$ and $\sigma = 0.1$, Monte Carlo simulations (with $10^8$ instances) provided on *stats stack exchange* `stats.stackexchange.com/q/535924` indicate that the approximations in Equation 3.34 and Equation 3.35 are tight, respectively, within factor $10^{-4}$ and factor $10^{-3}$.

When $\frac{1}{\mu_d} \neq 0$, $\frac{1}{\sqrt{n}}\frac{\sqrt{\mu_{d^2}}}{\mu_d}\sigma_{-1}^{\mathrm{dp}} \neq 0$, $\frac{1}{\mu_d} - 3\frac{1}{\sqrt{n}}\frac{\sqrt{\mu_{d^2}}}{\mu_d}\sigma_{-1}^{\mathrm{dp}} > 0$, with high probability,

$$\begin{aligned}
&\mathbb{E}\left[\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right] \quad \text{(by Eqn 3.34, Eqn 3.31, Eqn 3.32)}\\
&\quad\approx \frac{\sqrt{2}}{\sqrt{\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{-1}^{\mathrm{dp}})^2}} f_{\mathrm{daw}}\left(\frac{\frac{1}{\mu_d}}{\sqrt{2}\sqrt{\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{-1}^{\mathrm{dp}})^2}}\right)\\
&\quad= \sqrt{n}\frac{\sqrt{2}\mu_d}{\sqrt{\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}} f_{\mathrm{daw}}\left(\frac{\sqrt{n}}{\sqrt{2\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}}\right),
\end{aligned} \tag{3.36}$$

$$\begin{aligned}
&\mathbb{E}\left[\left(\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right)^2\right] \quad \text{(by Eqn 3.35, Eqn 3.31, Eqn 3.32)}\\
&\quad\approx \frac{1}{\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{-1}^{\mathrm{dp}})^2}\left(\frac{1}{\mu_d}\frac{\sqrt{2}}{\sqrt{\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{-1}^{\mathrm{dp}})^2}} f_{\mathrm{daw}}\left(\frac{\frac{1}{\mu_d}}{\sqrt{2}\sqrt{\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{-1}^{\mathrm{dp}})^2}}\right) - 1\right)\\
&\quad= n\frac{\mu_d^2}{\mu_{d^2}(\sigma_{-1}^{\mathrm{dp}})^2}\left(\sqrt{n}\frac{\sqrt{2}}{\sqrt{\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}} f_{\mathrm{daw}}\left(\frac{\sqrt{n}}{\sqrt{2\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}}\right) - 1\right).
\end{aligned} \tag{3.37}$$

We continue the derivation in Equation 3.30:

$$
\begin{aligned}
e_k^{\texttt{BCGo}} &= s_k^2 - 2s_k \frac{\mu_{d^k}}{\mu_d} \mathbb{E}\left[\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right] \quad \text{(by Eqn 3.31)} \\
&\quad + \left(\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{k-1}^{\mathrm{dp}})^2 + \frac{\mu_{d^k}^2}{\mu_d^2}\right) \mathbb{E}\left[\left(\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right)^2\right] \quad \text{(by Eqn 3.33)} \qquad (3.38) \\
&\approx s_k^2 - 2s_k \frac{\mu_{d^k}}{\mu_d}\sqrt{n}\frac{\sqrt{2}\mu_d}{\sqrt{\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}} f_{\mathrm{daw}}\left(\frac{\sqrt{n}}{\sqrt{2\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}}\right) \quad \text{(by Eqn 3.36, Eqn 3.37)} \\
&\quad + \left(\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{k-1}^{\mathrm{dp}})^2 + \frac{\mu_{d^k}^2}{\mu_d^2}\right) n\frac{\mu_d^2}{\mu_{d^2}(\sigma_{-1}^{\mathrm{dp}})^2}\left(\sqrt{n}\frac{\sqrt{2}}{\sqrt{\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}} f_{\mathrm{daw}}\left(\frac{\sqrt{n}}{\sqrt{2\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}}\right) - 1\right) \\
&= \mu_{d^k}^2 - 2\frac{\mu_{d^k}^2}{\mu_d}\sqrt{n}\frac{\sqrt{2}\mu_d}{\sqrt{\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}} f_{\mathrm{daw}}\left(\frac{\sqrt{n}}{\sqrt{2\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}}\right) \quad \text{(by Eqn 3.18)} \\
&\quad + \left(\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{k-1}^{\mathrm{dp}})^2 + \frac{\mu_{d^k}^2}{\mu_d^2}\right) n\frac{\mu_d^2}{\mu_{d^2}(\sigma_{-1}^{\mathrm{dp}})^2}\left(\sqrt{n}\frac{\sqrt{2}}{\sqrt{\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}} f_{\mathrm{daw}}\left(\frac{\sqrt{n}}{\sqrt{2\mu_{d^2}}\sigma_{-1}^{\mathrm{dp}}}\right) - 1\right). \qquad (3.39)
\end{aligned}
$$

The quantification of the approximation error is outside the scope of this work. □

We provide a theorem quantifying the expected error $e_k^{\texttt{BCGo}}$ (Equation 3.28) for the asymptotic case.

**Theorem 5.** *Let* $\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{-1}^{\mathrm{dp}})^2 \neq 0$ *and* $\frac{1}{\mu_d} - 3\frac{1}{\sqrt{n}}\frac{\sqrt{\mu_{d^2}}}{\mu_d}\sigma_{-1}^{\mathrm{dp}} > 0$*, where* $\mu_{d^k}$ *and* $\sigma_k^{\mathrm{dp}}$ *are given, respectively, in Equation 3.18 and Equation 3.15. Let us assume that* $\mathbb{E}\left[\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right] = \mu_d$ *and* $\mathrm{var}\left(\frac{1}{\frac{1}{n}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right) = \mathcal{O}(\frac{1}{n})$*. (We support the assumptions in the proof.) With high probability,*

$$
e_k^{\texttt{BCGo}} = \frac{\mu_{d^2}}{n}(\sigma_{k-1}^{\mathrm{dp}})^2 + \frac{\mu_{d^2}}{n}(\sigma_{k-1}^{\mathrm{dp}})^2 \mathrm{var}\left(\frac{1}{\frac{1}{n}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right) + \mu_{d^k}\mathrm{var}\left(\frac{1}{\frac{1}{n}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right), \qquad (3.40)
$$

*where* $D_{i,-1}^{\mathrm{dp}}$ *is given in Equation 3.14. This way,* $e_k^{\texttt{BCGo}} = \mathcal{O}\left(\frac{1}{n}\right)$.

*Proof.* When $\frac{1}{\sqrt{n}}\frac{\sqrt{\mu_{d^2}}}{\mu_d}\sigma_{-1}^{\mathrm{dp}} \neq 0$ and $\frac{1}{\mu_d} - 3\frac{1}{\sqrt{n}}\frac{\sqrt{\mu_{d^2}}}{\mu_d}\sigma_{-1}^{\mathrm{dp}} > 0$, we assume that

$$
\mathbb{E}\left[\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right] = \mu_d. \qquad (3.41)
$$

We support the aforementioned assumption by the result that $\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i d_i^{-1}} = \mu_d$ and by the evidence discussed in the proof of Theorem 4 that the approximation

(Equation 3.34) of the reciprocal of a Gaussian random variable with mean $\mu$ and variance $\sigma^2$ is tight (when $\mu \neq 0$, $\sigma \neq 0$, and $|\mu| - 3\sigma > 0$). This way,

$$\mathbb{E}\left[\left(\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right)^2\right] = \mu_d^2 + \mathrm{var}\left(\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right). \quad \text{(by Eqn 3.26, Eqn 3.41)} \tag{3.42}$$

We continue the derivation in Equation 3.38:

$$\begin{aligned} e_k^{\texttt{BCGo}} &= \mu_{d^k}^2 - 2\mu_{d^k}^2 + \left(\frac{1}{n}\frac{\mu_{d^2}}{\mu_d^2}(\sigma_{k-1}^{\mathrm{dp}})^2 + \frac{\mu_{d^k}^2}{\mu_d^2}\right)\left(\mu_d^2 + \mathrm{var}\left(\frac{1}{\frac{1}{n}\frac{1}{\mu_d}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right)\right) \\ &\quad \text{(by Eqn 3.18, Eqn 3.41, Eqn 3.42)} \\ &= \frac{\mu_{d^2}}{n}(\sigma_{k-1}^{\mathrm{dp}})^2 + \frac{\mu_{d^2}}{n}(\sigma_{k-1}^{\mathrm{dp}})^2 \mathrm{var}\left(\frac{1}{\frac{1}{n}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right) \\ &\quad + \mu_{d^k}\mathrm{var}\left(\frac{1}{\frac{1}{n}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right). \end{aligned} \tag{3.43}$$

We assume that $\mathrm{var}\left(\frac{1}{\frac{1}{n}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right) = \mathcal{O}\left(\frac{1}{n}\right)$, when $\frac{1}{\sqrt{n}}\frac{\sqrt{\mu_{d^2}}}{\mu_d}\sigma_{-1}^{\mathrm{dp}} \neq 0$ and $\frac{1}{\mu_d} - 3\frac{1}{\sqrt{n}}\frac{\sqrt{\mu_{d^2}}}{\mu_d}\sigma_{-1}^{\mathrm{dp}} > 0$. We support the aforementioned assumption by the conditions for the heuristic in Equation 3.35 and by the result that the variance of the squared average of $n$ independent and identically distributed random variables is $n$ times lower than the variance of a single such random variable (as suggested in Equation 3.25).

Hence, we conclude that $e_k^{\texttt{BCGo}} = \mathcal{O}(\frac{1}{n})$. □

We proceed by discussing the conditions when the average computed by `BCGo` results in less privatization noise compared to the average computed by `Cen`. We define the ratio between $e_k^{\texttt{BCGo}}$ and $e_k^{\texttt{Cen}}$ (given, respectively, by Equation 3.22 and Equation 3.40):

$$\frac{e_k^{\texttt{BCGo}}}{e_k^{\texttt{Cen}}} = \frac{\mu_{d^2}(\sigma_{k-1}^{\mathrm{dp}})^2}{(\sigma_k^{\mathrm{dp}})^2} + \frac{\mu_{d^2}(\sigma_{k-1}^{\mathrm{dp}})^2 \mathrm{var}\left(\frac{1}{\frac{1}{n}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right)}{(\sigma_k^{\mathrm{dp}})^2} + \frac{n\mu_{d^k}\mathrm{var}\left(\frac{1}{\frac{1}{n}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right)}{(\sigma_k^{\mathrm{dp}})^2}. \tag{3.44}$$

We assume that $\mathrm{var}\left(\frac{1}{\frac{1}{n}\sum_{i\in[n]} d_i D_{i,-1}^{\mathrm{dp}}}\right) = \mathcal{O}\left(\frac{1}{n}\right)$, when $\frac{1}{\sqrt{n}}\frac{\sqrt{\mu_{d^2}}}{\mu_d}\sigma_{-1}^{\mathrm{dp}} \neq 0$ and $\frac{1}{\mu_d} - 3\frac{1}{\sqrt{n}}\frac{\sqrt{\mu_{d^2}}}{\mu_d}\sigma_{-1}^{\mathrm{dp}} > 0$ (supported in the proof of Theorem 5). This way, the first term and the third term in Equation 3.44 are dominant when $n \to \infty$, i.e., the first term and the third term are $\mathcal{O}(1)$ whereas the second term is $\mathcal{O}(\frac{1}{n})$. We can expand the

first term in Equation 3.44:

$$\frac{\mu_{d^2}(\sigma_{k-1}^{\mathrm{dp}})^2}{(\sigma_k^{\mathrm{dp}})^2} = \mu_{d^2}\frac{2\log\left(\frac{5(m+2)}{4\delta'}\right)\left(\Delta_2(d_i^{k-1})\right)^2}{\left(\epsilon'/(m+2)\right)^2}\Bigg/\frac{2\log\left(\frac{5m}{4\delta'}\right)\left(\Delta_2(d_i^{k})\right)^2}{\left(\epsilon'/m\right)^2}$$
$$= \left(\frac{m}{m+2}\right)^2\frac{\log\left(\frac{5(m+2)}{4\delta'}\right)}{\log\left(\frac{5m}{4\delta'}\right)}\mu_{d^2}\left(\frac{\Delta_2(d_i^{k-1})}{\Delta_2(d_i^{k})}\right)^2. \tag{3.45}$$

We conjecture that, in the non-asymptotic case, $e_k^{\texttt{BCGo}} < e_k^{\texttt{Cen}}$ (Equation 3.44 is lower than 1) requires the following conditions: the number $m$ of sensitive attributes is high (Equation 3.45 is lower), the number $n$ of vertices is high (the second term and the third term in Equation 3.44 are lower), the degrees $\mathbf{d}$ are low ($\mu_{d^2}$ and $\mu_{d^k}$ in Equation 3.44 are lower), and $k \geq 2$ (but remains low due to $\mu_{d^k}$ in Equation 3.44) as, in Equation 3.45, we have

$$\frac{\Delta_2(d_i^{k-1})}{\Delta_2(d_i^{k})} = \frac{d_{\max}^{k-1}-(d_{\max}-1)^{k-1}}{d_{\max}^{k}-(d_{\max}-1)^{k}} \quad \text{(by Equation 3.16)}$$
$$< 1. \tag{3.46}$$

To summarize, we highlight that $\frac{\Delta_2(d_i^{k-1})}{\Delta_2(d_i^{k})} < 1$ suggests that we can have $e_k^{\texttt{BCGo}} < e_k^{\texttt{Cen}}$, i.e., the average computed by `BCGo` can have less privatization noise compared to the average computed by `Cen`.

**Empirical errors.** We define the empirical errors that correspond to the expected errors stated, respectively, in Theorem 3 and Theorem 4:

$$\hat{e}_k^{\texttt{Cen}} = \left(\frac{1}{n}\sum_{i\in[n]} d_i^k - \frac{1}{n}\sum_{i\in[n]} d_{i,k}^{\mathrm{dp}}\right)^2, \tag{3.47}$$

$$\hat{e}_k^{\texttt{BCGo}} = \left(\frac{1}{n}\sum_{i\in[n]} d_i^k - \frac{\texttt{SiGo}\left((d_{i,k-1}^{\mathrm{dp}})_{i=1}^n\right)}{\texttt{SiGo}\left((d_{i,-1}^{\mathrm{dp}})_{i=1}^n\right)}\right)^2, \tag{3.48}$$

where $d_{i,k}^{\mathrm{dp}}$ is given in Remark 1. We remark that the empirical errors are stated with the intention to evaluate the corresponding expected errors, which is done Subsection 3.3.7.

### 3.3.7 Experiments

In this subsection, we specify two sets of experiments and interpret the results.

**Experiment 3.1.** *We compare the accuracy of the averages of privatized attributes computed, respectively, by* `BCGo` *(Equation 3.9) and* `Cen`*. Secondly, we evaluate if the expected errors (Equation 3.22 and Equation 3.29) are tight (within factor of* 10 *to their empirical counterparts in Equation 3.47 and Equation 3.48) when* $n$ *ranges from* $10^2$ *to* $10^4$ *and* $\epsilon = 2^2$*. Thirdly, we evaluate if the expected errors decrease linearly in* $n$ *(as indicated by Theorem 5).*

**Experiment 3.2.** *We compare the utility of the regression models (Equation 3.10) fitted, respectively, using the averages computed by* `BCGo` *(Equation 3.9),* `SiGo` *(Algorithm 1), and* `Cen`*. In particular, we evaluate the MSE between true target values and predicted target values (of a test set of size* $10^3$*) when* $\epsilon$ *ranges from* $2^{-6}$ *to* $2^{10}$*. Usually,* $\epsilon > 10$ *is too high for privacy in practice [Hsu+14].*

We use a special case of the regression model given by Equation 3.10. That is, for each $i \in [n]$,

$$y_i = \theta_0 + \theta_1 d_i^{-1} + \theta_2 d_i^{1/2} + \theta_3 d_i^2 + \xi_i^{\text{reg}}, \tag{3.49}$$

where $\theta_1$ is coupled with a decreasing feature, $\theta_2$ is coupled with a slower than linearly increasing feature, and $\theta_3$ is coupled with a quicker than linearly increasing feature (this covers the three sensitivities in Equation 3.16), and where regression noise $\xi_i^{\text{reg}}$ is drawn independently from $\mathcal{N}(0,1)$. We fix the regression parameters to $\theta_0 = \theta_1 = \theta_2 = \theta_3 = 1$ to generate synthetic regression datasets. We fix the regularization parameter to $\lambda = 1$.

Our synthetic graph dataset is created by generating a power-law degree sequence with the shape parameter fixed to $\gamma = 2$ (discussed in Section A.1) and generating a graph from the configuration model (Definition 4). For real graph datasets, we use the graphs of the email-Eu-core network dataset (1005 vertices and 25571 edges) and the autonomous systems AS-733 dataset (6474 vertices and 13895 edges), both of which are part of SNAP [LK14]. We have processed the real graph datasets by Step 3 of the procedure described in Section A.1 and have removed self-loops.

We precise the experiment setup:

- We fix $\delta = \frac{1}{n^2}$, where such choice is also used in Near et al. [Nea+19] We fix the lowest degree to $d_{\min} = 3$ and the highest degree to $d_{\max} = 10^2$. We fix the number of gossip iterations to $\mathrm{t}_{\text{go}} = 2^{10}$.

- In Experiment 3.1, we normalize the errors dividing them by $(\sigma_k^{\text{dp}})^2$ (Equation 3.15). In Experiment 3.2, we normalize the MSE between true target values and predicted target values dividing it by $(\sigma_y)^2$, where $\sigma_y$ is the standard deviation of true target values, similarly as in Gupta and Kling [GK11].

- We fix the number of experiment repetitions to $\mathrm{t}_{\text{exp}} = 2^{10}$. For the synthetic dataset, in each experiment repetition we generate a degree sequence and a graph. For Experiment 3.2, in each experiment repetition we generate privatized features and target values for fitting, and a test set of size $10^3$ which contains target values assigned to an independently generated synthetic graph of order $10^3$ for evaluating the real graph datasets. In Experiment 3.1, each privatized attribute $d_{i,-1'}^{\text{dp}}$ (Remark 1) is bounded (Equation 3.17) so that the conditions for the expected error (Equation 3.29) hold. This way, $\sigma_{-1}^{\text{dp}}$ in Equation 3.29 is substituted with an approximated variance (discussed in Section A.2) of a bounded $d_{i,-1'}^{\text{dp}}$. In Experiment 3.2, the privatized attributes are bounded to improve the utility of the regression models fitted, respectively, using the averages computed by `BCGo`, `SiGo`, and `Cen`.

- We fix the significance level $\alpha_{\text{ci}} = 0.05$ (the computation of confidence intervals is described in Equation 2.5).

In Section A.4, we discuss secondary experiments.

**Results.** Figure 3.1 illustrates the results of Experiment 3.1 on the synthetic graph dataset, which indicate that `BCGo` results in more noisy averages than `Cen` when $k = 1/2$, and in less noisy averages when $k = 2$. (The case with $k = -1$ is in Section A.3.) Upon $k = 2$, the reason of `BCGo` outperforming `Cen` is explained by the ratio in Equation 3.46 being sufficiently lower than 1.

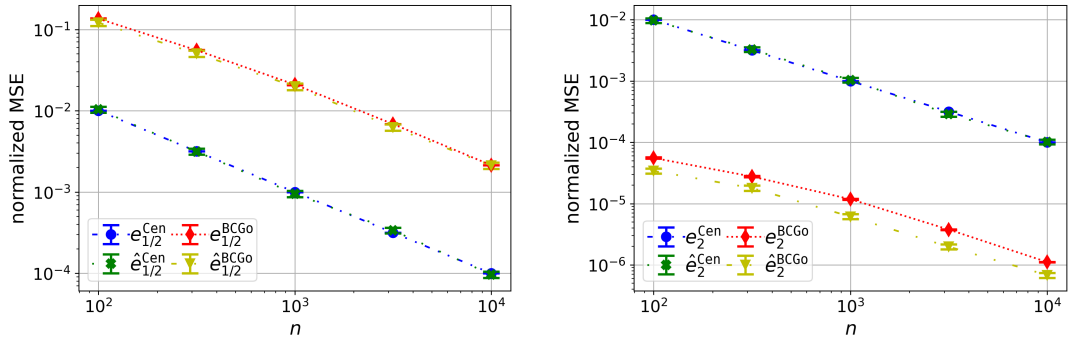


Figure 3.1: Comparison of the expected errors (Equation 3.22 and Equation 3.29) and the empirical errors (Equation 3.47 and Equation 3.48), when $\epsilon = 2^2$. $k = 1/2$ (left) and $k = 2$ (right) refer, respectively, to estimation of $\mu_{d^{1/2}}$ and $\mu_{d^2}$ (present in Equation 3.12).

Figure 3.2 illustrates the results of Experiment 3.2 on the synthetic dataset, which indicate that `BCGo` performs better than `Cen` and `SiGo`. Essentially, `BCGo` requires lower privatization noise for averaging the feature $d_i^2$ (in Equation 3.49) compared to `Cen` and `SiGo`. We remark that `SiGo` performs relatively well because the elements of each average (Equation 3.12 and Equation 3.13) for regularized least squares are scaled by the same factor $\frac{d_i}{\mu_d}$ (as in Theorem 2), which leads to relatively accurate parameter estimates. The quantification of the bias in the parameter estimates computed by `SiGo` is outside the scope of this work.

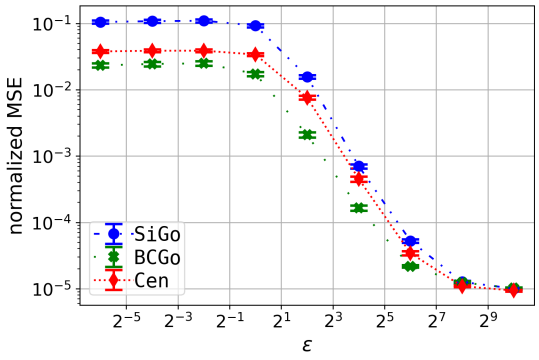


Figure 3.2: Comparison of the MSE between true target values and predicted target values (Equation 3.49), when averaging, respectively, by `BCGo`, `SiGo`, and `Cen`, and when $n = 10^3$

Figure 3.3 illustrates the results of Experiment 3.2 on the real graph datasets, which indicate that `BCGo` performs better than `Cen` and `SiGo`, except on the email network dataset upon a higher privacy budget. `BCGo` performs worse upon a higher privacy budget because the estimation of $\mu_y$ involves privatization (unlike `Cen` and `SiGo`) and depends on $\mu_{d^2}$ and $\mu_{d^k}$ (Equation 3.44), where $\mu_{d^2}$ and $\mu_{d^k}$ are higher for the email network dataset compared to the autonomous systems dataset.

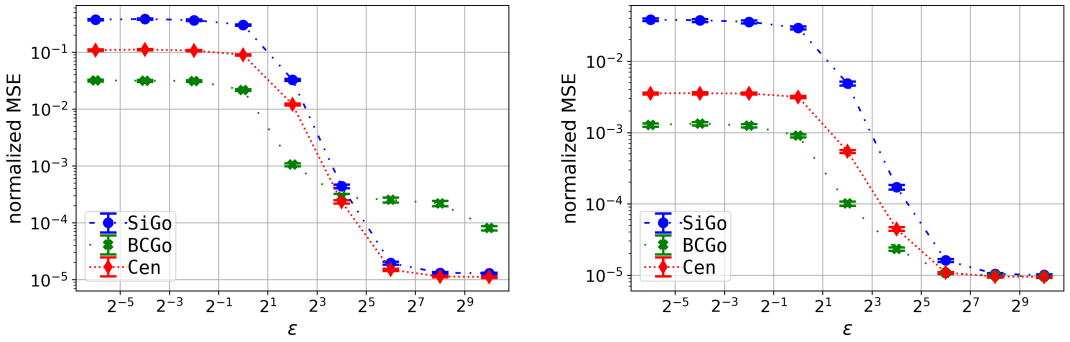


Figure 3.3: Comparison of the MSE between true target values and predicted target values (Equation 3.49), when averaging, respectively, by `BCGo`, `SiGo`, and `Cen`. We use the email network dataset (left) and the autonomous systems dataset (right).

The experiments were run on a machine with an Intel E5-2643 processor (3.30 GHz) and 192 GB of RAM; for the synthetic dataset, this took 3 hours. Similarly, for the email network dataset this took 1 hour and for the autonomous systems dataset this took 9 hours. The most time consuming operation is the matrix power in Algorithm 1. The storage is mostly affected by the adjacency matrix of the graph; we remark that in our implementation we store the adjacency matrix in a sparse matrix. Our implementation is inappropriate when $n \geq 10^5$ because Algorithm 1 is a centralized simulation of distributed averaging. For the study of Algorithm 1 on graphs whose order is $n \geq 10^5$, we suggest implementing Algorithm 1 in a distributed manner.

### 3.3.8 Conclusion

We have proved an asymptotic guarantee (Theorem 5) for the MSE between the average of sensitive attributes and the average of locally privatized attributes computed by the bias-correcting gossip algorithm (Equation 3.9), when the graph of agents is modeled by the configuration model (Definition 4), when the degrees of agents are sensitive, and when the agents interact without handshakes.

Equation 3.44 indicates that distributed averaging performed by the bias-correcting gossip algorithm `BCGo` can result in less privatization noise in comparison to the averaging function $f(\mathbf{x}) = \frac{1}{n}\sum_{i\in[n]} x_i$ (where $\mathbf{x} \in \mathbb{R}^n$), when we compute U-statistics (Definition 1) of degree 1 from locally privatized degrees raised to a power $k = 2$, and when the graph has low degrees (i.e., when the 2-nd raw moment $\mu_{d^2}$ and the $k$-th raw moment $\mu_{d^k}$ of the degrees in Equation 3.44 are low).

We remark that the privatization of degrees by the Gaussian mechanism (Definition 6) is suboptimal since degrees are natural numbers and the Gaussian mechanism

results in real-valued privatization noise. A more suitable privatization mechanism could be one that results in privatization noise with a discrete support, e.g., the discrete Gaussian mechanism [CKS20] or the tailored noise mechanism discussed in Chapter 4.

**Future work.** Let $\mathbf{x} \in \mathbb{R}^n$. The unbiased sample variance of $\mathbf{x}$ is defined as follows:

$$\begin{aligned} \bar{\sigma}_x^2 &= \frac{1}{n(n-1)} \sum_{j>i} (x_i - x_j)^2 \\ &= \frac{1}{n-1} \sum_{i \in [n]} (x_i - \mu_x)^2. \end{aligned} \tag{3.50}$$

We highlight that `BCGo` can compute $\mu_x$ and $\frac{1}{n}\sum_{i\in[n]}(x_i-\mu_x)^2$ which are U-statistics of degree 1. This way, it remains to identify a strategy to obtain $n-1$ so that `BCGo` could be used to to compute $\bar{\sigma}_x^2$ which is a U-statistic of degree 2. Taking the example with the unbiased sample variance, we conjecture that, by mathematical induction, it can be shown that `BCGo` can compute U-statistics of arbitrary degree.

# Chapter 4

# Tailored noise mechanism

In this chapter, we continue to study the setting of privacy-preserving distributed averaging where each agent of a communication network is attributed a vector of sensitive attributes, each agent locally privatizes each sensitive attribute, and each agent intends to collaboratively compute the averages over the privatized attributes. In particular, we focus on a pre-processing step that happens prior to privatization, where the central curator solves a convex program and obtains privatization functions for privatizing sensitive attributes. Essentially, such privatization functions correspond to a utility-maximizing privatization mechanism which is shared to each agent, so that each agent could locally privatize its sensitive attributes.

To elaborate on our setting, we consider a group of agents, where each agent computes features (for fitting regression models) by transforming sensitive attributes, and where the transformations have high-magnitude gradients or singularities. In such setting, there is a risk of obtaining an outlier feature due to transforming a privatized attribute in an interval where the transformation has high-magnitude gradients. We mitigate the aforementioned risk by providing a tailored noise mechanism for privatizing features by solving a convex program in such a way that (i) only informative intervals of transformations are selected, (ii) the variance of privatization noise is minimized, and (iii) the biasedness of privatization noise is minimized.

We highlight that this chapter is based on a collaborative work with *Moitree Basu* who at the time was a doctorate student. Our common work mainly involved the following axes:

- Axis 1: Expression of models whose parameters can be computed from U-statistics (provided in Equation 4.1 below)
- Axis 2: Discretization of domains of features (discussed in Subsection 4.4.3)
- Axis 3: Expression of the constraints and the objective function of the convex program for obtaining privatization functions (discussed, respectively, in Subsection 4.4.1 and Subsection 4.4.2)
- Axis 4: Experiment setup (discussed in Section 4.6)

Regarding our individual theses, Moitree Basu had focused on the generalized expression of models mentioned in Axis 1 above. While for myself, I had focused on

the comparison of the model where privatization noise is added before transforming attributes to features and the model where attributes are transformed to features before adding privatization noise (as such question arose while working on the bias-correcting gossip algorithm discussed in Section 3.3).

## 4.1 Introduction

We consider a setting where a group of agents intends to collaboratively fit a statistical model by sharing locally privatized features. For fitting a statistical model, one of the objectives is to identify such features which result in an accurate statistical model. In this work, we consider that features can be computed from attributes or other features, and such computations can be transformations with singularities. For example, the reciprocal function $f(x) = \frac{1}{x}$ has a singularity at $x = 0$. When privatizing features using a classic mechanism, e.g., the Gaussian mechanism (Definition 6) or the Laplace mechanism (Definition 7), we tend to firstly transform sensitive attributes into features, and only then add privatization noise. Otherwise, when privatization noise is added on sensitive attributes first, the classic mechanisms might result in a noisy value that falls too close to the singularity of a transformation, and thus the transformation results in an outlier feature. In other words, a privatized feature is susceptible to fall in an interval where the transformation has high-magnitude gradients. Furthermore, the classic mechanisms are suboptimal in terms of utility as they add, typically, more noise than necessary, as discussed in Balle and Wang [BW18].

In this work, we provide a tailored noise mechanism for privatizing features by solving a convex program in such a way that (i) only informative intervals of transformations are selected, (ii) the variance of privatization noise is minimized, and (iii) the biasedness of privatization noise is minimized. The execution of the tailored noise mechanism (`TNM`) firstly requires solving a convex program. We assume that the constraint solver is executed by a central curator, and the central curator shares the result of the constraint solver with each agent.

Regarding the motivation of the studied problem, `TNM` produces discrete probability distributions (as opposed to continuous) for privatizing features (an extension of `TNM` for continuous probability distributions is outside the scope of this work). This is similar to the discrete Gaussian mechanism [CKS20] which addresses the case when the data is discrete and spans over integers. Furthermore, certain classes of data have a bounded domain (e.g., for count data, we have natural numbers where the lowest value is 0), thus it might be useful to have the domain of a privatized feature, which is bounded in the same way as the domain of a respective sensitive feature. We remark that the Gaussian mechanism (Definition 6) and the Laplace mechanism (`Laplace`) result in unbounded domains of privatized features.

Regarding the novel qualities of `TNM`, we apply discretization strategies that mitigate the issue of obtaining an outlier feature due to transforming a privatized attribute in an interval where the transformation has high-magnitude gradients (or singularities). Also, we provide a convex program whose solution consists of a tailored privatization function parametrized by a sensitive feature, where the variance of the tailored privatization function is minimized and the absolute

difference (biasedness) between the mean of the tailored privatization function and the sensitive feature is minimized.

We show on synthetic datasets and a synthetically-extended real dataset that `TNM` results, typically, in a lower MSE between true target values and predicted target values compared to `Laplace`, when fitting a linear regression model by regularized least squares that use U-statistics (Definition 1), and when the feature function is either the logarithm, the reciprocal, or the tangent.

**Outline.** In Section 4.2, we precise our setting. In Section 4.3, we perform the literature study. In Section 4.4, we design `TNM`. In Section 4.5, we describe the implementation of `TNM` using the `cvxopt` library for convex programs. In Section 4.6, we discuss the experiments. In Section 4.7, we conclude.

## 4.2 Preliminaries

In this section, we provide and recall the notation, and then we precise the setting.

**Communication model.** We have a group of agents each of which is attributed a data tuple with a scalar target value and a sensitive feature vector from a space $\mathcal{X} \subseteq \mathbb{R}^{\tilde{m}}$, where $\tilde{m} \in \mathbb{N}_1$ is the number of sensitive features.

In this work, the term "sensitive feature" encapsulates sensitive attributes. In particular, we consider that features can be computed from attributes or other features, and such computations can be transformations with singularities. We refer to the functions used to compute features from attributes as feature functions (e.g., we have the logarithm function, the identity function, the exponential function).

**Threat model.** We have a central curator who computes conditional probability mass functions (to which we refer as privatization functions) and shares them with the agents, so that each agent can hide its sensitive features under privatization noise (local differential privacy). The space of a privatized feature is denoted by $\tilde{\mathcal{X}}$ (we remark that $\tilde{\mathcal{X}} \supseteq \mathcal{X}$, where $\mathcal{X}$ is the space of sensitive features). This way, for each $j \in [\tilde{m}]$, $x \in \mathcal{X}_j$, $\tilde{x} \in \tilde{\mathcal{X}}_j$, we have a privatization function with a conditional probability $p_j(\tilde{x} \,|\, x)$, where $\mathcal{X}_j$ and $\tilde{\mathcal{X}}_j$ denote, respectively, the domain of the $j$-th sensitive feature and the domain of the $j$-th privatized feature.

**Statistical model.** We assume that our statistical model is a linear function $h^*$ that maps a feature vector to a scalar target value, where $h^*$ is parametrized by $\theta^* \in \mathbb{R}^{m+1}$, and where $m \in \mathbb{N}_1$ is the number of features. Once each agent privatizes its features and they are shared with the central curator, then the central curator can compute a matrix $\mathbf{X} \in \mathbb{R}^{n \times (m+1)}$ composed of features and target values over all agents, where $n$ is the number of agents. Then, the central curator fits a statistical model $h$ (a linear function that maps a feature vector to a target value) using a function $g : \mathbf{X} \mapsto \hat{\theta}$, where $h$ is parametrized by $\hat{\theta}$. In this work, we aim for obtaining such privatization functions that minimize an objective value $\mathcal{L}(\hat{\theta}, \theta^*)$, where $\mathcal{L}$ is an objective function whose common choice is the sum of the squared differences between the elements of the estimated parameters $\hat{\theta}$ (computed from privatized features) and the true parameters $\theta^*$. We remark that the bolded font denotes vectors and matrices, as indicated in Table 2.1.

We consider the number $l \in \mathbb{N}_1$ of models/approaches for obtaining parameter estimates $\hat{\theta}$. That is, for each $k \in [l]$, we have a function $g_k : \mathbf{X} \mapsto \hat{\theta}$ (to which we

refer as Model $k$). We remark that distinct choices of $g_k$ include distinct fitting methods or distinct privatization strategies. In particular, we consider regularized least squares that uses U-statistics as indicated by Equation 2.9. This way, we have

$$g_k\Big(f_{k,1}(\mathbf{X}), f_{k,2}(\mathbf{X}), \ldots, f_{k,q_k}(\mathbf{X})\Big), \tag{4.1}$$

where $q_k \in \mathbb{N}$ is the number of U-statistics (Definition 1) for fitting Model $k$, $g_k : \mathbb{R}^{q_k} \to \mathbb{R}^{m+1}$ is a function for computing model parameters from U-statistics, and, for each $s \in [q_k]$, we have a function $f_{k,s} : \mathbb{R}^{n\times(m+1)} \to \mathbb{R}$ which computes a U-statistic from $\mathbf{X}$.

In this work, the central curator computes the privatization functions by solving a convex program. Since convex programs are solved quicker (and require less memory) upon less constraint variables and constraints, the sensitive features of the agents are discretized. That is, the discretization is performed using thresholds, where, for every two adjacent thresholds, we attribute a representative value to which a sensitive feature is mapped to if the sensitive feature is in between those two adjacent thresholds (we perform binning, where the bins are the same for each feature vector of each agent). This way, a discretized domain $\mathcal{X}_{k,j}$ includes the values of a sensitive feature $j$ and Model $k$, and a discretized domain $\tilde{\mathcal{X}}_{k,j} \supseteq \mathcal{X}_{k,j}$ includes the values of a privatized feature $j$ and Model $k$.

In summary, for computing privatization functions, the central curator solves a convex program to obtain a privatization function with a conditional probability $p_{k,j}(\tilde{x} \,|\, x)$, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}, \tilde{x} \in \tilde{\mathcal{X}}_{k,j}$, where $\tilde{m}_k$ is the number of sensitive features used for fitting Model $k$.

In Table 4.1, we summarize the notation related to sensitive features and privatized features.

Table 4.1: Notation related to sensitive features and privatized features

| Notation | Meaning |
|---|---|
| $\mathcal{X}_{k,j}$ | Discretized domain of a sensitive feature $j$ for Model $k$ |
| $\tilde{\mathcal{X}}_{k,j}$ | Discretized domain of a privatized feature $j$ for Model $k$ |
| $x$ | Value of a sensitive feature |
| $\tilde{x}$ | Value of a privatized feature |

We briefly expand on the notation. We use the acronym `dp` to refer to the application of a privatization function for differential privacy noise. We use the acronym `tns` to refer to the feature function that transforms attributes into features. We use the symbol "$\circ$" to denote the function composition. For example, for functions $f, g, h$, one of their compositions is expressed as $h \circ g \circ f$, which corresponds to firstly applying $f$, then $g$, and finally $h$.

In Table 4.2, we summarize the notation related to the work on tailored privatization.

Table 4.2: Notation related to the work on tailored privatization

| Notation | Meaning |
|---|---|
| $n$ | Number of agents |
| $m$ | Number of features |
| $m'$ | Number of attributes ($m' \leq m$) |
| $\mathbf{X}$ | Matrix of features and target values |
| $\theta^*$ | True parameters of a statistical model |
| $\hat{\theta}$ | Estimated parameters of a statistical model |
| $l$ | Number of models for estimating parameters of a statistical model |
| $\tilde{m}_k$ | Number of sensitive features for Model $k$ |
| $q_k$ | Number of U-statistics for Model $k$ |
| $g_k$ | Function computing parameters of Model $k$ from U-statistics |
| $f_{k,s}$ | Function computing a U-statistic $s$ for Model $k$ from $\mathbf{X}$ |
| $\mathcal{L}$ | Objective function |

Finally, we remark that we particularly focus on the following two privatization strategies (which are models/approaches for obtaining parameter estimates $\hat{\theta}$):

- Model `tns` $\circ$ `dp`. In this model, differential privacy noise is added to attributes first, and only then the attributes are transformed into features.
- Model `dp` $\circ$ `tns`. In this model, attributes are transformed into features first, and only then differential privacy noise is added to the features.

## 4.3 Literature study

In this section, we perform the literature study.

We highlight that this work has been mainly inspired by the utility-maximizing privatization mechanisms proposed by Ghosh et al. [GRS12] and Cormode et al. [CKS17; CKS21] (the works are similar though Cormode et al. [CKS17; CKS21] have added additional constraints to resolve some anomalies in the work of Ghosh et al. [GRS12]). The two utility-maximizing privatization mechanisms are characterized by solving a convex program to obtain a privatization function for count data. On one hand, our work is similar to the two mechanisms in the sense that we also define a convex program. On the other hand, our work is different in the sense that we consider a broader class of data than counts. Also, we have an additional constraint for unbiasedness. Furthermore, we take additional care of the singularities upon the computation of features.

Brenner and Nissim [BN10] indicates that it is easier to provide stronger theoretical guarantees for maximizing the utility of privatized counts than for maximizing the utility of privatized sums. However, in our work, we focus on utility in practice as opposed to theoretical guarantees.

The issue with the privatization mechanisms that do not rely on convex programs (e.g., the Gaussian mechanism, `Laplace` or the discrete Gaussian mechanism

[CKS20]) is that they result in suboptimal utility in exchange for simplicity of implementation and application. We remark that, upon defining a convex program, we have a discrete privatization function where we can explicitly specify the differential privacy requirement between every two of its probabilities (similarly as in Definition 5). Further, there exist approaches where privatization functions are obtained using generative adversarial networks [RCP20] or stochastic gradient descent [Wan+21]. However, those techniques require a training set whereas in our approach we avoid such dependency.

We discuss several other mechanisms that are based on convex programs. Gupte and Sundararajan [GS10] is similar to Ghosh et al. [GRS12] in the sense that the authors consider count data though they use a different objective function. Geng and Viswanath [GV16] considers central differential privacy (where we trust a central curator and share with it the true values of sensitive features). Shokri [Sho15] considers local differential privacy, however the work does not consider unbiasedness.

We remark that blowfish privacy [HMD14] is a generalization of differential privacy and is appropriate for cases when the trade-off between privacy and utility is of interest. However, such framework requires background knowledge on the graph of agents.

Finally, we remark that Mironov et al. [Mir12] discusses a privatization issue of `Laplace` due to the least significant bits. In our case, such issue is avoided by discretizing the features to the bins that do not rely on the least significant bits (e.g., the representative values of the bins can be rounded to a higher significant bit).

## 4.4 Approach

In this section, we intend to define a convex program for computing the conditional probabilities of a privatization function which is tailored for the following conditions:

1. Sampling from the privatization function guarantees $(\epsilon, 0)$-differential privacy (Definition 5).

2. The biasedness of the privatization function is minimized, i.e., the absolute difference between the mean of the privatization function and the sensitive feature parametrizing the privatization function is minimized.

3. The variance of the privatization function is minimized.

4. Each conditional probability $p_{k,j}(\tilde{x} \mid x)$ of the privatization function must sum to 1, while each such probability must be greater or equal to 0 and less than or equal to 1.

We remark that a successful solution of the aforementioned convex program is interpreted as `TNM` because the solution includes the privatization functions for privatizing features.

In the subsections that follow, we elaborate how `TNM` integrates the four requirements stated in the aforementioned list. In Subsection 4.4.1, we define the

constraints. In Subsection 4.4.2, we define the objective function. In Subsection 4.4.3, we review some strategies for discretization of domains of sensitive features and domains of privatized features.

### 4.4.1 Definition of the constraints

In this subsection, we define the constraints to cover Requirement 1 (for ensuring differential privacy) and Requirement 4 (for ensuring valid probability mass functions), and to partially cover Requirement 2 (for ensuring bias minimization). The complete coverage of Requirement 2 is achieved in conjunction with the objective function discussed later in Subsection 4.4.2.

**Differential privacy constraint.** Since, for Model $k$, we have $\tilde{m}_k$ sensitive features, we split the total privacy budget $\epsilon$ evenly for each sensitive feature. That is, for Model $k$, an even split of the total privacy budget $\epsilon$ is the following:

$$\epsilon_k = \frac{\epsilon}{\tilde{m}_k}.$$

Let $\epsilon \geq 0$. For local $\epsilon$-differential privacy (similarly as in Definition 5), it is requires that, for each $k \in [l], j \in [\tilde{m}_k], x_1, x_2 \in \mathcal{X}_{k,j}, T \subseteq \tilde{\mathcal{X}}_{k,j}$, we have

$$\sum_{\tilde{x} \in T} p_{k,j}(\tilde{x} \,|\, x_1) \leq e^{\epsilon_k} \sum_{\tilde{x} \in T} p_{k,j}(\tilde{x} \,|\, x_2), \tag{4.2}$$

which is a linear inequality constraint with the constraint variables $p_{k,j}(\tilde{x} \,|\, x_1)$ and $p_{k,j}(\tilde{x} \,|\, x_2)$. We remark that $x_1$ and $x_2$ in Equation 4.2 are adjacent values under the notion of local differential privacy, where in local differential privacy we add noise, typically, directly to sensitive attributes as opposed to statistics computed from them. Also, we remark that the sums in Equation 4.2 are due to the addition law of probability, as we intend to privatize a discrete class of data.

Since the number of subsets $T \subseteq \tilde{\mathcal{X}}_{k,j}$ is exponential in $|\tilde{\mathcal{X}}_{k,j}|$ (results in a large number of constraints), we reformulate Equation 4.2. That is, for each $k \in [l], j \in [\tilde{m}_k], x_1, x_2 \in \mathcal{X}_{k,j}, \tilde{x} \in \tilde{\mathcal{X}}_{k,j}$, we have

$$p_{k,j}(\tilde{x} \,|\, x_1) \leq e^{\epsilon_k} p_{k,j}(\tilde{x} \,|\, x_2). \tag{4.3}$$

We can further reduce the number of constraints by reformulating Equation 4.3. That is, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}, \tilde{x} \in \tilde{\mathcal{X}}_{k,j}$, we have

$$p_{k,j}(\tilde{x} \,|\, x) \leq e^{\epsilon_k} p^{\max}_{k,j}(\tilde{x}), \tag{4.4}$$

$$p^{\max}_{k,j}(\tilde{x}) \leq e^{\epsilon_k} p_{k,j}(\tilde{x} \,|\, x), \tag{4.5}$$

where $p^{\max}_{k,j}(\tilde{x})$ is a constraint variable for the highest value in $\{p_{k,j}(\tilde{x} \,|\, x) : x \in \mathcal{X}_{k,j}\}$.

Finally, we rewrite Equation 4.4 and Equation 4.5 to the normal form of constraints (i.e., the terms with constraint variables are moved to the left hand side of the "less than or equal" symbol). That is, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}, \tilde{x} \in \tilde{\mathcal{X}}_{k,j}$, we have

$$p_{k,j}(\tilde{x} \,|\, x) - e^{\epsilon_k} p^{\max}_{k,j}(\tilde{x}) \leq 0, \tag{4.6}$$

$$p^{\max}_{k,j}(\tilde{x}) - e^{\epsilon_k} p_{k,j}(\tilde{x} \,|\, x) \leq 0. \tag{4.7}$$

We remark that a constraint solver might encounter numerical difficulties when the coefficients of constraint variables are high. This way, we scale Equation 4.6 and Equation 4.7. That is, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}, \tilde{x} \in \tilde{\mathcal{X}}_{k,j}$, we have

$$\frac{1}{e^{\epsilon_k}} p_{k,j}(\tilde{x} \mid x) - p_{k,j}^{\max}(\tilde{x}) \leq 0, \tag{4.8}$$

$$\frac{1}{e^{\epsilon_k}} p_{k,j}^{\max}(\tilde{x}) - p_{k,j}(\tilde{x} \mid x) \leq 0. \tag{4.9}$$

Hence, for ensuring local $\epsilon$-differential privacy, we have two distinct linear inequality constraints.

**Bias minimization constraint.** We design the bias minimization of the privatization functions by firstly defining a linear inequality constraint. That is, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}$, we have

$$\sum_{\tilde{x} \in \tilde{\mathcal{X}}_{k,j}} \tilde{x} p_{k,j}(\tilde{x} \mid x) \leq x + c_{k,j,x}, \tag{4.10}$$

where $c_{k,j,x}$ is a constraint variable that quantifies the bias (ideally equals 0). We remark that we use the "less than or equal" symbol as opposed to equality because we aim for a larger solution space.

We rewrite Equation 4.10 in the normal form. That is, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}$, we have

$$\sum_{\tilde{x} \in \tilde{\mathcal{X}}_{k,j}} \tilde{x} p_{k,j}(\tilde{x} \mid x) - c_{k,j,x} \leq x. \tag{4.11}$$

Finally, we scale Equation 4.11. That is, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}$, we have

$$\frac{1}{\max_{\tilde{x} \in \tilde{\mathcal{X}}_{k,j}} |\tilde{x}|} \left( \sum_{\tilde{x} \in \tilde{\mathcal{X}}_{k,j}} \tilde{x} p_{k,j}(\tilde{x} \mid x) - c_{k,j,x} \right) \leq \frac{x}{\max_{\tilde{x} \in \tilde{\mathcal{X}}_{k,j}} |\tilde{x}|}. \tag{4.12}$$

Hence, for ensuring bias minimization, we have one linear inequality constraint. However, we remind that up until now we have discussed only the first part of the bias minimization, as the second part of the bias minimization is part of the objective function (discussed later in Subsection 4.4.2).

**Constraints of probability mass functions.** The probabilities of a probability mass function must sum to 1, while each such probability is greater or equal to 0 and less than or equal to 1.

We define the constraint for the probabilities of a probability mass function summing to 1. That is, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}$, we have

$$\sum_{\tilde{x} \in \tilde{\mathcal{X}}_{k,j}} p_{k,j}(\tilde{x} \mid x) = 1, \tag{4.13}$$

which is a linear equality constraint in its normal form.

We define the constraint so that each probability is higher or equal to 0 (Equation 4.13 already implies that no probability is higher than 1 because their sum equals to 1). That is, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}, \tilde{x} \in \tilde{\mathcal{X}}_{k,j}$, we have

$$p_{k,j}(\tilde{x} \mid x) \geq 0, \tag{4.14}$$

which is a linear inequality constraint.

We rewrite Equation 4.14 in the normal form. That is, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}, \tilde{x} \in \tilde{\mathcal{X}}_{k,j}$, we have

$$-p_{k,j}(\tilde{x} \mid x) \leq 0. \tag{4.15}$$

We require that the constraint variable $p^{\max}_{k,j}(\tilde{x})$ is less than or equal to 1. That is, for each $k \in [l], j \in [\tilde{m}_k], \tilde{x} \in \tilde{\mathcal{X}}_{k,j}$, we have

$$p^{\max}_{k,j}(\tilde{x}) \leq 1, \tag{4.16}$$

which is a linear inequality constraint in its normal form.

Further, we require that the constraint variable $p^{\max}_{k,j}(\tilde{x})$ is greater or equal to the probabilities in $\{p_{k,j}(\tilde{x} \,|\, x) : x \in \mathcal{X}_{k,j}\}$. That is, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}, \tilde{x} \in \tilde{\mathcal{X}}_{k,j}$, we have

$$p_{k,j}(\tilde{x} \,|\, x) \leq p^{\max}_{k,j}(\tilde{x}), \tag{4.17}$$

which is a linear inequality constraint.

We rewrite Equation 4.17 in its normal form. That is, for each $k \in [l], j \in [\tilde{m}_k], x \in \mathcal{X}_{k,j}, \tilde{x} \in \tilde{\mathcal{X}}_{k,j}$, we have

$$p_{k,j}(\tilde{x} \,|\, x) - p^{\max}_{k,j}(\tilde{x}) \leq 0. \tag{4.18}$$

We remark that if the constraint in Equation 4.18 is met, it guarantees that the constraint in Equation 4.8 is met also.

Hence, for valid probability mass functions, we have one linear equality constraint and three linear inequality constraints.

### 4.4.2 Definition of the objective function

In this subsection, we define the objective function so that it covers Requirement 3 (for ensuring variance minimization). We highlight that the objective function completes the coverage of Requirement 2 (for ensuring bias minimization) whose initial part is covered in Subsection 4.4.1 above.

Our intention is to minimize the total objective function $\mathcal{L}$ by minimizing the objective function $\mathcal{L}_k$ of Model $k$, for each $k \in [l]$, i.e., we minimize

$$\mathcal{L} = \sum_{k \in [l]} \mathcal{L}_k. \tag{4.19}$$

We define the objective function of Model $k$ as

$$\mathcal{L}_k = \mathcal{L}^{\text{vce}}_k + \mathcal{L}^{\text{bias}}_k, \tag{4.20}$$

where $\mathcal{L}_k^{\text{vce}}$ is the objective value for minimizing variance and $\mathcal{L}_k^{\text{bias}}$ is the objective value for minimizing bias.

**Variance minimization.** In Model `tns` $\circ$ `dp`, we firstly privatize attributes and then transform them into features for computing U-statistics. For this reason, in the objective function, we have the variance of attributes and the variance of features. Thus, for each $j \in [\tilde{m}_k]$, let $r_j \in \mathbb{N}_1$ denote the number of features computed from an attribute $j$. Then, for each $t \in [r_j]$, let $f'_{k,j,t}$ denote the $t$-th feature function.

We define the objective function for variance minimization as follows:

$$\mathcal{L}_k^{\text{vce}} = \sum_{j\in[\tilde{m}_k],t\in[r_j],x\in\mathcal{X}_{k,j},\tilde{x}\in\tilde{\mathcal{X}}_{k,j}} \rho_{k,j,t}^{\text{vce}} p_{k,j}(\tilde{x} \mid x) \left(f'_{k,j,t}(\tilde{x}) - f'_{k,j,t}(x)\right)^2, \tag{4.21}$$

where

$$\rho_{k,j,t}^{\text{vce}} = 10^{-2} \min\left(1, \frac{1}{\max_{x\in\mathcal{X}_{k,j},\tilde{x}\in\tilde{\mathcal{X}}_{k,j}} \left(f'_{k,j,t}(\tilde{x}) - f'_{k,j,t}(x)\right)^2}\right) \tag{4.22}$$

is the scaling term of the objective function for variance minimization. We remark that the term $10^{-2}$ is there due to the constraint variable $p_{k,j}(\tilde{x} \mid x)$ belonging to an interval of length 1 (thus the partial derivatives of the objective function for variable minimization should not be too high to avoid proposals that are outside that interval).

We conclude that the objective function for variance minimization is linear with respect to the constraint variables.

**Bias minimization.** We define the objective function for bias minimization as follows:

$$\mathcal{L}_k^{\text{bias}} = \sum_{j\in[\tilde{m}_k],x\in\mathcal{X}_{k,j}} \rho_{k,j}^{\text{bias}} c_{k,j,x}^2, \tag{4.23}$$

where

$$\rho_{k,j}^{\text{bias}} = 10^{-2} \frac{1}{\max_{\tilde{x}\in\tilde{\mathcal{X}}_{k,j}} \tilde{x}^2} \tag{4.24}$$

is the scaling term of the objective function for bias minimization. We remark that the term $10^{-2}$ is there so that the objective function for bias minimization has a lower influence on the total objective function.

We conclude that the objective function for bias minimization is convex since the constraint variable $c_{k,j,x}$ is squared.

### 4.4.3 Discretization of domains of features

In this subsection, we describe some strategies for discretization of domains of sensitive features and domains of privatized features, which are based on thresholds and representative values. That is, for every two adjacent thresholds, we attribute a representative value to which a sensitive feature is mapped to if the sensitive feature is in between those two adjacent thresholds.

We recall two classic discretization strategies: equal distance discretization and equal frequency discretization, which are discussed in Zheng and Casari [ZC18].

In equal distance discretization, we choose the number of thresholds, compute the difference between the highest and the lowest value, and determine the length of a discretization subinterval (we refer to it by a bin) by dividing the previously computed difference by the number of thresholds. Finally, we place thresholds at the boundaries of each bin and fix the representative value in the middle of two adjacent thresholds.

Equal distance discretization can be also used in the sense of mapping features to the transformation scale (e.g., the natural logarithm scale), then computing the thresholds and the representative values, and finally remapping them to the original scale (for the natural logarithm scale, such remapping is $e^x$). This results in more (or less) bins in the intervals where the gradient of the transformation of features has a higher (or lower) magnitude. We remark that the magnitude of a gradient can get high when the transformation has singularities. For example, the singularity in $f(x) = \frac{1}{x}$ is at $x = 0$.

In equal frequency discretization, we determine the locations of the thresholds in such a way that there would be an approximately equal number of features falling in between every two adjacent thresholds. We fix each representative value in the middle of two adjacent thresholds.

## 4.5 Implementation

In this section, we relate our constraint problem (discussed in Section 4.4) to the standard form accepted by the `cvxopt` package (discussed in Subsection 2.6.1). Firstly, we precise the order and counts of the constraint variables in our convex program Then, we precise the order and counts of the constraints. Finally, we elaborate on the implementation details of the objective function.

Even though the convex program in Section 4.4 is quadratic (it has a quadratic objective function and linear constraints), we formulate it as a more general convex program. This way, future extensions that result in convex constraints or a convex objective function would be more straightforward to integrate. We remark that, in this work, we already discuss the scaling of constraints and the scaling of the objective function, which are typical practices to reduce the numerical difficulty for solving convex programs.

We remark that, in this work, we have used the default constraint solver of `cvxopt`, which is based on Cholesky decomposition. This choice is faster compared to an alternative that is based on the LDL decomposition, which is more suitable for convex programs that are numerically difficult, e.g., when it is difficult to appropriately scale the constraints or the objective function.

**Order of constraint variables**. Let $\text{idx}(\cdot)$ denote a map from a value to its index in its discrete domain. In Table 4.3, we list the constraint variables according to their order in the convex program and we also provide a formula for the index of a particular constraint variable.

Table 4.3: Order and indices of constraint variables in the convex program

| Order | Variable | Index |
|---|---|---|
| 1 | $p_{k,j}(\tilde{x} \mid x)$ | $\sum_{k' \in [k]} \sum_{j' \in [\tilde{m}_{k'}]} \lvert\mathcal{X}_{k',j'}\rvert\lvert\tilde{\mathcal{X}}_{k',j'}\rvert + \sum_{j' \in [j]} \lvert\mathcal{X}_{k,j'}\rvert\lvert\tilde{\mathcal{X}}_{k,j'}\rvert$ $+\text{idx}(x)\lvert\tilde{\mathcal{X}}_{k,j}\rvert + \text{idx}(\tilde{x})$ |
| 2 | $p^{\max}_{k,j}(\tilde{x})$ | $\sum_{k' \in [k]} \sum_{j' \in [\tilde{m}_{k'}]} \lvert\tilde{\mathcal{X}}_{k',j'}\rvert + \sum_{j' \in [j]} \lvert\tilde{\mathcal{X}}_{k,j'}\rvert + \text{idx}(\tilde{x})$ |
| 3 | $c_{k,j,x}$ | $\sum_{k' \in [k]} \sum_{j' \in [\tilde{m}_{k'}]} \lvert\mathcal{X}_{k',j'}\rvert + \sum_{j' \in [j]} \lvert\mathcal{X}_{k,j'}\rvert + \text{idx}(x)$ |

In Table 4.4, we provide a formula for the total number of constraint variables.

Table 4.4: Total number of constraint variables.

| Variable | Total number | Total number (in experiments) |
|---|---|---|
| $p_{k,j}(\tilde{x} \mid x)$ | $\sum_{k \in [l], j \in [\tilde{m}_k]} \lvert\mathcal{X}_{k,j}\rvert\lvert\tilde{\mathcal{X}}_{k,j}\rvert$ | $(m' + m)\lvert\mathcal{X}\rvert\lvert\tilde{\mathcal{X}}\rvert$ |
| $p^{\max}_{k,j}(\tilde{x})$ | $\sum_{k \in [l], j \in [\tilde{m}_k]} \lvert\tilde{\mathcal{X}}_{k,j}\rvert$ | $(m' + m)\lvert\tilde{\mathcal{X}}\rvert$ |
| $c_{k,j,x}$ | $\sum_{k \in [l], j \in [\tilde{m}_k]} \lvert\mathcal{X}_{k,j}\rvert$ | $(m' + m)\lvert\mathcal{X}\rvert$ |

**Order of constraints.** In Table 4.5 and Table 4.6 below, we list, respectively, the order of linear equality constraints and the order of linear inequality constraints in the convex program.

Table 4.5: Order of linear equality constraints

| Order | Description | Reference |
|---|---|---|
| 1 | Probabilities sum to 1 | Equation 4.13 |

Table 4.6: Order of linear inequality constraints

| Order | Description | Reference |
|---|---|---|
| 1 | Differential privacy | Equation 4.8 |
| 2 | Probabilities are greater than 0 | Equation 4.15 |
| 3 | Highest probabilities are less than or equal to 1 | Equation 4.18 |

We remind that Equation 4.18 implies Equation 4.9, where the latter equation is the remaining constraint for differential privacy.

In Table 4.7, we provide a formula for the total number of each type of constraints.

Table 4.7: Total number of constraints. In our experiments, we use discretized domains of equal size.

| Reference | Total number | Total number (experiments) |
|---|---|---|
| Equation 4.13 | $\sum_{k\in[l],j\in[\tilde{m}_k]} \lvert\mathcal{X}_{k,j}\rvert$ | $(m'+m)\lvert\mathcal{X}\rvert$ |
| Equation 4.8 | $\sum_{k\in[l],j\in[\tilde{m}_k]} \lvert\mathcal{X}_{k,j}\rvert\lvert\tilde{\mathcal{X}}_{k,j}\rvert$ | $(m'+m)\lvert\mathcal{X}\rvert\lvert\tilde{\mathcal{X}}\rvert$ |
| Equation 4.15 | $\sum_{k\in[l],j\in[\tilde{m}_k]} \lvert\mathcal{X}_{k,j}\rvert\lvert\tilde{\mathcal{X}}_{k,j}\rvert$ | $(m'+m)\lvert\mathcal{X}\rvert\lvert\tilde{\mathcal{X}}\rvert$ |
| Equation 4.18 | $\sum_{k\in[l],j\in[\tilde{m}_k]} \lvert\mathcal{X}_{k,j}\rvert\lvert\tilde{\mathcal{X}}_{k,j}\rvert$ | $(m'+m)\lvert\mathcal{X}\rvert\lvert\tilde{\mathcal{X}}\rvert$ |

**Implementation of the objective function.** Regarding our objective function $\mathcal{L}$ (defined in Equation 4.19), we remind that it is composed of the objective function $\mathcal{L}_k^{\text{vce}}$ (Equation 4.21) for minimizing variance and the objective function $\mathcal{L}_k^{\text{bias}}$ (Equation 4.23) for minimizing bias. This way, in each iteration, the constraint solver evaluates the objective function using the values of the constraint variables with probabilities of privatization functions, i.e., $p_{k,j}(\tilde{x} \mid x)$, and the values of the constraint variables indicating the amount of biasedness, i.e., $c_{k,j,x}$.

Regarding the initial proposal of the solution of the constraint solver, we fix the constraint variables related to probabilities, i.e., $p_{k,j}(\tilde{x} \mid x)$ and $p_{k,j}^{\max}(\tilde{x})$, so that they correspond to uniform distributions. Then, we fix the constraint variables $c_{k,j,x}$ so that the bias minimization constraints result in equalities.

## 4.6 Experiments

In this section, we discuss the experiments.

In Subsection 4.6.1, we describe datasets. In Subsection 4.6.2, we precise the experiment setup. In Subsection 4.6.3, we discuss the results of the experiments. In Subsection 4.6.4, we discuss secondary experiments.

### 4.6.1 Dataset description

In this subsection, we define the linear regression model and describe our datasets.

**Linear regression model.** Let $m \in \mathbb{N}_1$. For our experiments, we define a multiple linear regression model with $m+1$ regression parameters and a scalar target value. For each $i \in [n]$, we have

$$y_i = \theta_0 + \theta_1 x_{i,1} + \ldots + \theta_m x_{i,m} + \xi_i^{\text{reg}}, \tag{4.25}$$

where $x_{i,1}, x_{i,2}, \ldots, x_{i,m}$ are (sensitive) features, $\theta_0, \theta_1, \ldots, \theta_m \in \mathbb{R}$ are regression parameters, $\xi_i^{\text{reg}}$ is regression noise which is an independent observation of $\mathcal{N}(0, \sigma^2)$, and $\sigma$ is the standard deviation of regression noise.

We leave the following remarks on the privacy guarantees for locally privatizing the aforementioned sensitive features $x_{i,1}, x_{i,2}, \ldots, x_{i,m}$:

- Upon `Laplace` (Definition 7), we assume that the lowest value of each sensitive feature and the highest value of each sensitive feature are known. This way,

each agent can compute the $l_1$ sensitivity (Equation 2.14) of `Laplace`. We recall that `Laplace` ensures $\epsilon$-differential privacy.

- Upon `TNM`, we assume that the lowest value of each sensitive feature and the highest value of each sensitive feature are known. This way, equal distance discretization (which is required for obtaining privatization functions and is discussed in Subsection 4.4.3) allows for each agent to keep each of its sensitive feature hidden because upon the aforementioned assumption the central curator can perform equal distance discretization without knowing any of the sensitive features. However, for equal frequency discretization, the central curator would need to know each sensitive feature.

- Upon `TNM`, local privatization of sensitive features is $\epsilon$-differentially private due to Equation 4.6 and Equation 4.7, which are derived from Equation 4.2. For privatizing its sensitive feature, each agent would firstly discretize it using the same discretization strategy as the central curator. We argue that the privacy guarantees of `TNM` (upon equal distance discretization) and `Laplace` are comparable because we rely on the same assumptions. Yet, the discretization in `TNM` results in a loss of utility due to the presence of granularity.

- Referring to Sei and Ohsuga [SO22], we assume that target values are non-sensitive. Even though a target value $y_i$ (Equation 4.25) is a linear combination of sensitive features, the exact values of the regression parameters and the regression noise remain unknown, thus the values of the sensitive features can not be deduced.

- Referring to the composition theorem in Dwork and Roth [DR14], privatization of the sensitive features upon even splits of the total privacy budget $\epsilon$ lead to $\epsilon$-differential privacy.

We fit the regression model defined in Equation 4.25 by regularized least squares (discussed in Subsection 2.4.1). That is, referring to Equation 2.9, we compute the vector $\hat{\theta}$ of parameter estimates as follows:

$$\hat{\theta} = \left(\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_{m+1}\right)^{-1}\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{y}, \tag{4.26}$$

where

$$\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X} = \begin{bmatrix} 1 & \frac{1}{n}\sum_{i\in[n]} x_{i,1} & \dots & \frac{1}{n}\sum_{i\in[n]} x_{i,m} \\ \frac{1}{n}\sum_{i\in[n]} x_{i,1} & \frac{1}{n}\sum_{i\in[n]} x_{i,1}^2 & \ddots & \frac{1}{n}\sum_{i\in[n]} x_{i,1}x_{i,m} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{1}{n}\sum_{i\in[n]} x_{i,m} & \frac{1}{n}\sum_{i\in[n]} x_{i,1}x_{i,m} & \dots & \frac{1}{n}\sum_{i\in[n]} x_{i,m}^2 \end{bmatrix}$$

$$\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{y} = \begin{bmatrix}\frac{1}{n}\sum_{i\in[n]} y_i & \frac{1}{n}\sum_{i\in[n]} y_i x_{i,1} & \dots & \frac{1}{n}\sum_{i\in[n]} y_i x_{i,m}\end{bmatrix}^{\mathrm{T}},$$

and where $\lambda \in \mathbb{R}$ is the regularization parameter and $\mathbf{I}$ is the $(m+1)\times(m+1)$ identity matrix.

**Datasets.** We describe four synthetic datasets ds1, ds2a, ds2b, ds3, and a synthetically-extended real dataset misra1d (the size of the real dataset was extremely small, thus we generated additional instances). Let $\epsilon^* = 10^{-2}$ be the closest distance to which an attribute can reach the singularity when transforming the attribute into a feature. We list the descriptions of the datasets:

- ds1. We have the regression model so that, for each $i \in [n]$, a target value $y_i = \theta_0 + \theta_1 \log(a_i) + \xi_i^{\text{reg}}$, where an attribute $a_i$ is drawn independently from $\text{uni}(\epsilon^*, 1)$, regression noise $\xi_i^{\text{reg}}$ is drawn independently from $\mathcal{N}(0, 1)$, true regression parameters $\theta_0 = \theta_1 = 1$. We intend to obtain regression parameter estimates $\hat{\theta}_0$ and $\hat{\theta}_1$ so that a predicted target value $y_i^{\text{pred}} = \hat{\theta}_0 + \hat{\theta}_1 \log(a_i)$.

- ds2a. We have the regression model so that, for each $i \in [n]$, a target value $y_i = \theta_0 + \theta_1 \frac{1}{a_i} + \xi_i^{\text{reg}}$, where an attribute $a_i$ is drawn independently from $\text{uni}(\epsilon^*, 1)$, regression noise $\xi_i^{\text{reg}}$ is drawn independently from $\mathcal{N}(0, 1)$, true regression parameters $\theta_0 = \theta_1 = 1$. We intend to obtain regression parameter estimates $\hat{\theta}_0$ and $\hat{\theta}_1$ so that a predicted target value $y_i^{\text{pred}} = \hat{\theta}_0 + \hat{\theta}_1 \frac{1}{a_i}$.

- ds2b. We have the regression model so that, for each $i \in [n]$, a target value $y_i = \theta_0 + \theta_1 a_i + \theta_2 \frac{1}{a_i} + \xi_i^{\text{reg}}$, where an attribute $a_i$ is drawn independently from $\text{uni}(\epsilon^*, 1)$, regression noise $\xi_i^{\text{reg}}$ is drawn independently from $\mathcal{N}(0, 1)$, true regression parameters $\theta_0 = \theta_1 = \theta_2 = 1$. We intend to obtain regression parameter estimates $\hat{\theta}_0$, $\hat{\theta}_1$, and $\hat{\theta}_2$ so that a predicted target value $y_i^{\text{pred}} = \hat{\theta}_0 + \hat{\theta}_1 a_i + \hat{\theta}_2 \frac{1}{a_i}$.

- ds3. We have the regression model so that, for each $i \in [n]$, a target value $y_i = \theta_0 + \theta_1 \tan(a_i) + \xi_i^{\text{reg}}$, where an attribute $a_i$ is drawn independently from $\text{uni}(-\frac{\pi}{2} + \epsilon^*, \frac{\pi}{2} - \epsilon^*)$, regression noise $\xi_i^{\text{reg}}$ is drawn independently from $\mathcal{N}(0, 1)$, true regression parameters $\theta_0 = \theta_1 = 1$. We intend to obtain regression parameter estimates $\hat{\theta}_0$ and $\hat{\theta}_1$ so that a predicted target value $y_i^{\text{pred}} = \hat{\theta}_0 + \hat{\theta}_1 \tan(a_i)$.

- misra1d. We have the regression model so that, for each $i \in [n]$, a target value $y_i = \frac{\hat{\theta}_1^{\mu} \hat{\theta}_2^{\mu} a_i}{1 + \hat{\theta}_2^{\mu} a_i} + \xi_i^{\text{reg}}$, where an attribute $a_i = 10^3 \times a_i'$, $a_i'$ is drawn independently from $\text{uni}(0, 1)$, regression noise $\xi_i^{\text{reg}} = 6.85 \times 10^{-2} \times \xi_i^{\text{reg}\prime}$, $\xi_i^{\text{reg}\prime}$ is drawn independently from $\mathcal{N}(0, 1)$, regression parameters $\hat{\theta}_1^{\mu} = 4.37 \times 10^2$ and $\hat{\theta}_2^{\mu} = 3.02 \times 10^{-4}$. We intend to obtain regression parameter estimates $\hat{\theta}_0$, $\hat{\theta}_1$, and $\hat{\theta}_2$ so that a predicted target value $y_i^{\text{pred}} = \hat{\theta}_0 + \hat{\theta}_1 \frac{1}{1 + c_1 a_i} + \hat{\theta}_2 \frac{1}{1 + c_2 a_i}$, where $c_1 = \hat{\theta}_2^{\mu} + \hat{\theta}_2^{\sigma}$, $c_2 = \hat{\theta}_2^{\mu} + 1.1 \times \hat{\theta}_2^{\sigma}$, and the sample standard deviation (of $\hat{\theta}_2^{\mu}$) $\hat{\theta}_2^{\sigma} = 2.93 \times 10^{-6}$. We remark that the coefficient $6.85 \times 10^{-2}$ of $\xi_i^{\text{reg}}$ and values $\hat{\theta}_1^{\mu}$, $\hat{\theta}_2^{\mu}$, $\hat{\theta}_2^{\sigma}$ are taken from itl.nist.gov/div898/strd/nls/data/LINKS/v-misra1d.shtml, whereas the coefficient $10^3$ of $a_i$ is the order of magnitude of the distance between the lowest and the highest target values in the original dataset.

Regarding the misra1d dataset, we have approximated its non-linear regression model, i.e., $y_i = \frac{\hat{\theta}_1^{\mu} \hat{\theta}_2^{\mu} a_i}{1 + \hat{\theta}_2^{\mu} a_i} + \xi_i^{\text{reg}}$ by a linear one, i.e., $y_i \approx \hat{\theta}_0 + \hat{\theta}_1 \frac{1}{1 + c_1 a_i} + \hat{\theta}_2 \frac{1}{1 + c_2 a_i}$.

That is, for $\alpha_1, \alpha_2 \in \mathbb{R}$, we have

$$\begin{aligned}\frac{\hat{\theta}_1^\mu \hat{\theta}_2^\mu a_i}{1+\hat{\theta}_2^\mu a_i} &= \hat{\theta}_1^\mu \left(1 - \frac{1}{1+\hat{\theta}_2^\mu a_i}\right) \\ &\approx \hat{\theta}_1^\mu \left(1 - \sum_{i\in[2]} \frac{\alpha_i}{1+c_i a_i}\right) \\ &= \hat{\theta}_0 + \hat{\theta}_1 \frac{1}{1+c_1 a_i} + \hat{\theta}_2 \frac{1}{1+c_2 a_i}. \end{aligned} \tag{4.27}$$

For a closer approximation of the non-linear regression model, the sum should be over more than two values (contrary to the case in Equation 4.27).

In Section 3.3 on averaging on arbitrary graphs, we have a regression model (Equation 3.10) with multiple features that can involve the reciprocal transformation. We remark that this motivates the applicability of the aforementioned dataset ds2b.

### 4.6.2 Experiment setup

In this subsection, we specify two sets of experiments.

**Experiment 4.1.** *We compare the utility (MSE between true target values and predicted target values) of the regression models (Equation 4.25) fitted, respectively, by non-privatized features, features privatized by* `Laplace`*, and features privatized by* `TNM`*, when there is one feature computed from one attribute.*

**Experiment 4.2.** *We compare the utility (MSE between true target values and predicted target values) of the regression models (Equation 4.25) fitted, respectively, by non-privatized features, features privatized by* `Laplace`*, and features privatized by* `TNM`*, when there are two features computed from one attribute.*

In both experiments, we compare Model `tns` $\circ$ `dp` (where attributes are privatized first and then the privatized attributes are transformed into features) to Model `dp` $\circ$ `tns` (where attributes are transformed into features first and then the features are privatized).

We precise the experiment setup:

- We discretize domains of sensitive features and domains of privatized features in equal distance and 20 bins (respectively, $|X| = 20$ and $|\tilde{X}| = 20$).
- We use $\epsilon \in \{2^{-1}, 2^0, 2^{1/2}, 2^1, 2^{3/2}, 2^2, 2^{5/2}, 2^3, 2^{7/2}, 2^4, 2^5, 2^6\}$ for evaluating the privacy budget. We recall that, usually, $\epsilon > 10$ is too high for privacy in practice [Hsu+14]. We fix the number of agents to $n = 10^4$.
- We fix the tolerance of the constraint solver to $10^{-7}$. The solution of constraint solver is successful when primal infeasibility (pres), dual infeasibility (dres), and gap are lower than the fixed tolerance. For more details, we refer to cvxopt.org/userguide/solvers. (In experiment results, we indicate pres, dres, and gap which are the highest over the evaluated privacy budget). We fix the maximal number of iterations of the constraint solver to $2^7$.

- We normalize the MSE between true target values and predicted target values dividing it by $(\sigma_y)^2$, where $\sigma_y$ is the standard deviation of true target values. (We remark that such normalization is discussed in more detail in Gupta and Kling [GK11].) We perform cross-validation with 10 folds. We fix the regularization parameter to $\lambda = 10^3$.

- We fix the number of experiment repetitions to $t_{exp} = 2^7$. In each experiment repetition, we privatize features. Each synthetic dataset is generated once for all experiment repetitions.

- We fix the significance level to $\alpha_{ci} = 0.05$ (the computation of confidence intervals is described in Equation 2.5).

### 4.6.3 Result interpretation

In this subsection, we discuss the results of the two sets of experiments.

Figure 4.1 illustrates the results of Experiment 4.1 on ds1. Due to sufficiently fine discretization, `TNM` performs better than `Laplace`. In `Laplace`, Model `tns` ∘ `dp` performs slightly better than Model `dp` ∘ `tns` upon a lower privacy budget and slightly worse upon a higher privacy budget. We conclude that the two models perform similarly because the log transformation approaches its singularity relatively slowly. Similarly, in `TNM`, Model `tns` ∘ `dp` performs similar to Model `dp` ∘ `tns`.

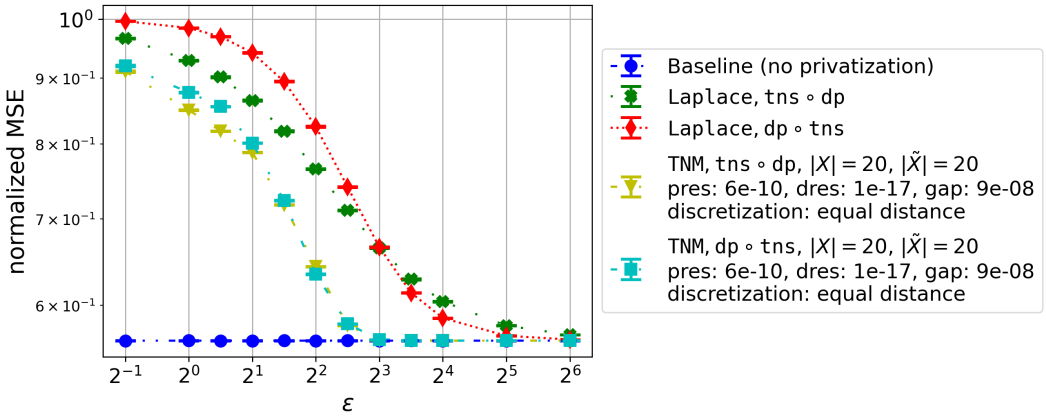


Figure 4.1: Experiment 4.1 on ds1. (In the convex program, the discretized domain for Model `tns` ∘ `dp` of `TNM` is in the logarithm scale.)

Figure 4.2 illustrates the results of Experiment 4.1 on ds2a. Due to sufficiently fine discretization, `TNM` performs better than `Laplace`. In `Laplace`, Model `dp` ∘ `tns` performs better than Model `tns` ∘ `dp` because the reciprocal transformation approaches its singularity relatively quickly, which results in a higher risk of privatized attributes falling into an interval close to the singularity. Similarly, in `TNM`, Model `dp` ∘ `tns` performs better than Model `tns` ∘ `dp`.

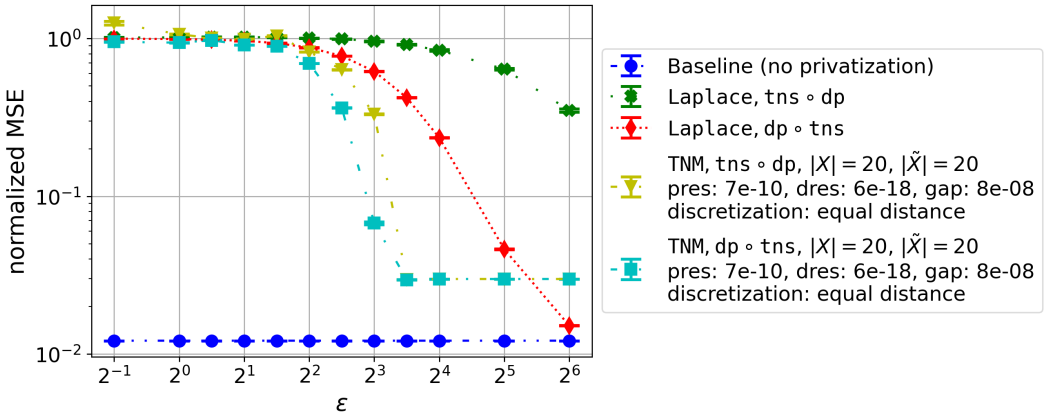


Figure 4.2: Experiment 4.1 on ds2a. (In the convex program, the discretized domain for Model `tns` ∘ `dp` of `TNM` is in the reciprocal scale.)

Figure 4.3 illustrates the results of Experiment 4.1 on ds3. Due to sufficiently fine discretization, `TNM` performs better than `Laplace`. In `Laplace`, Model `dp` ∘ `tns` performs better than Model `tns` ∘ `dp` because the tangent transformation approaches its singularity relatively quickly. In `TNM`, Model `tns` ∘ `dp` performs slightly worse than Model `dp` ∘ `tns` upon a lower privacy budget.

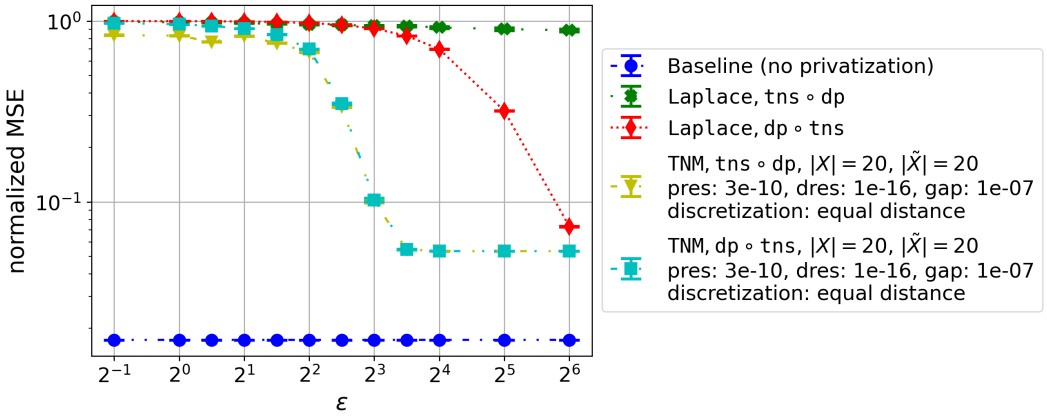


Figure 4.3: Experiment 4.1 on ds3. (In the convex program, the discretized domain for Model `tns` ∘ `dp` of `TNM` is in the tangent scale.)

Figure 4.4 illustrates the results of Experiment 4.2 on ds2b. Due to sufficiently fine discretization, `TNM` performs better than `Laplace`. In `Laplace`, Model `dp` ∘ `tns` performs better than Model `tns` ∘ `dp` because the reciprocal transformation approaches its singularity relatively quickly. However, in `TNM`, a higher privacy budget for one feature was enough for Model `tns` ∘ `dp` to outperform Model `dp` ∘ `tns` (contrary to Figure 4.2, where the privacy budget is same for both models).

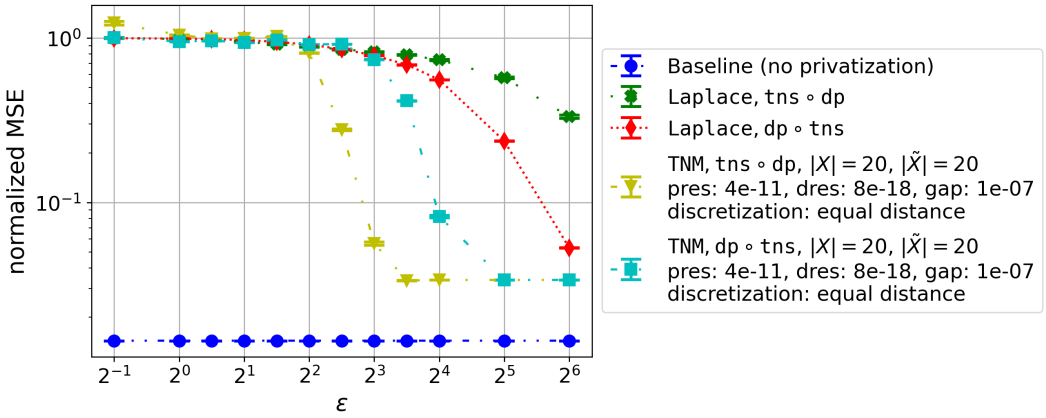


Figure 4.4: Experiment 4.2 on ds2b. (In the convex program, the discretized domain for Model `tns` ∘ `dp` of `TNM` is in the reciprocal scale.)

Figure 4.5 illustrates the results of Experiment 4.2 on misra1d. Due to splitting the privacy budget in Model `dp` ∘ `tns` and the transformations in Equation 4.27 staying relatively far from their singularities, `TNM` performs better than `Laplace` only for Model `tns` ∘ `dp`. In `Laplace`, Model `tns` ∘ `dp` performs better than Model `dp` ∘ `tns` upon a higher privacy budget and performs worse upon a lower privacy budget. In `TNM`, Model `tns` ∘ `dp` performs better than Model `dp` ∘ `tns` because, in Model `tns` ∘ `dp`, the privacy budget remains unsplit.

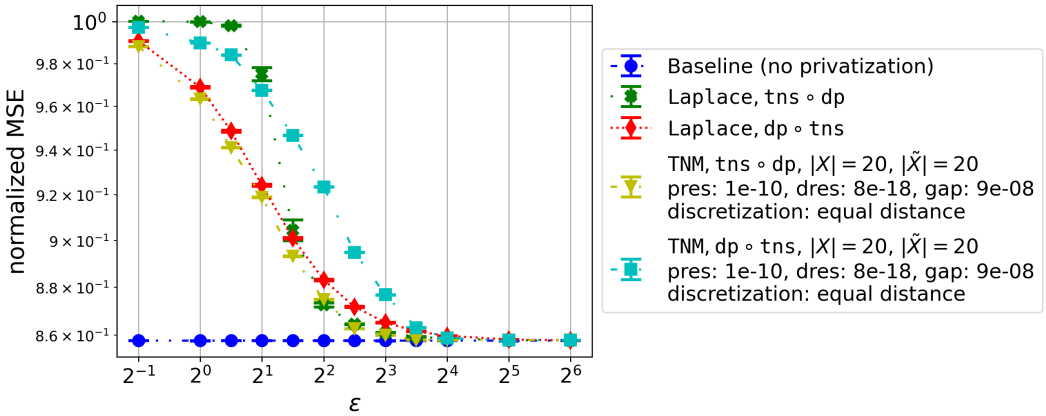


Figure 4.5: Experiment 4.2 on misra1d

The experiments were run on a machine with an Intel E5-2643 processor (3.30 GHz) and 192 GB of RAM; this took 15 minutes. For ds1, ds2a, and ds3, we had 880 constraint variables (Table 4.4) and 2440 constraints (Table 4.7). For ds2b and misra1d, we had 1320 constraint variables and 3660 constraints. For ds1, the memory requirement excessed our resources when the discretization granularity reached 100 bins (the implementation attempted to allocate space for 20400 constraint variables and 60200 constraints).

### 4.6.4 Secondary experiments

In this subsection, we discuss secondary experiments. In particular, we compare equal distance discretization with equal frequency discretization. Also, we compare coarser and finer domains of sensitive features and domains of privatized features.

Figure 4.6 illustrates a comparison between equal distance discretization and equal frequency discretization on ds1. The two discretization strategies result in similar utility because the two discretization strategies result in similar threshhold placements (the logarithm transformation approaches its singularity relatively slowly).

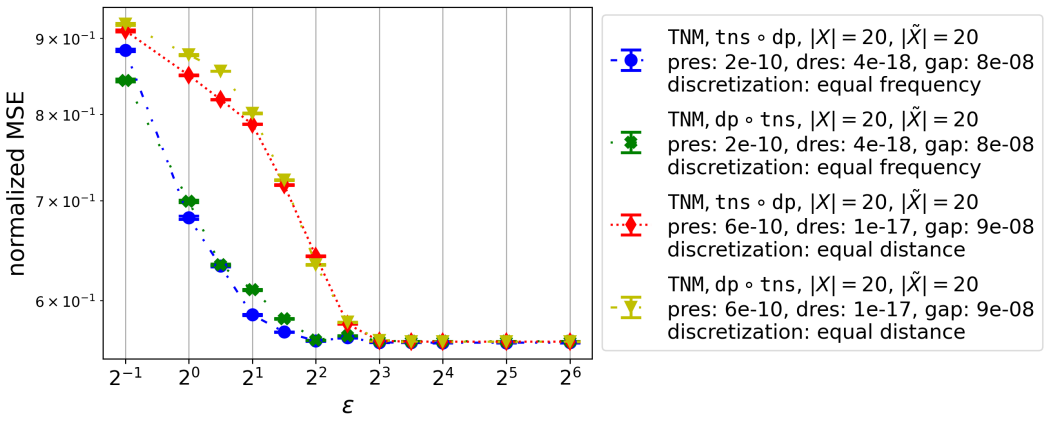


Figure 4.6: Comparison between equal distance discretization and equal frequency discretization on ds1

Figure 4.7 illustrates a comparison between equal distance discretization and equal frequency discretization on ds2a. Equal frequency discretization fails to result in gradual improvement of utility over the evaluated privacy budget because it is not fine enough in the interval with high-magnitude gradients (the reciprocal transformation approaches its singularity relatively quickly).

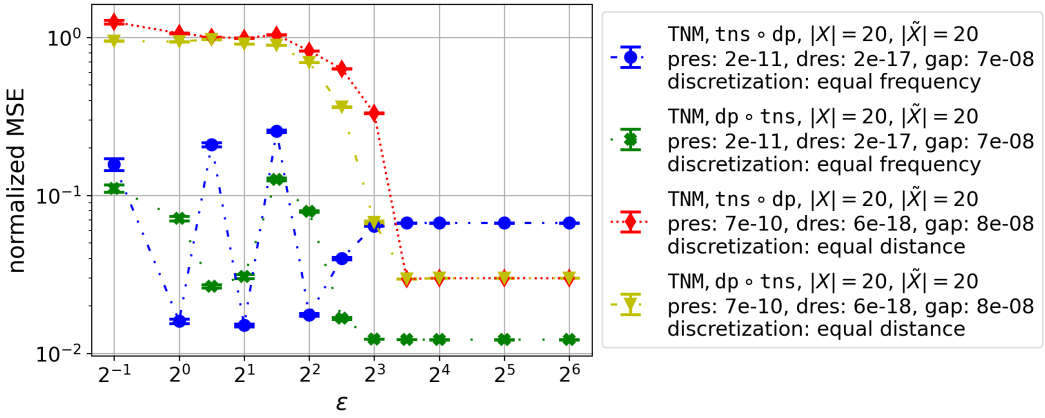


Figure 4.7: Comparison between equal distance discretization and equal frequency discretization on ds2a

Figure 4.8 illustrates a comparison between a finer discretization and a coarser discretization of domains of sensitive features on ds2a upon equal distance discretization. We observe that a finer discretization of sensitive features results in an improvement in utility for both Model `tns` ∘ `dp` and Model `dp` ∘ `tns`.

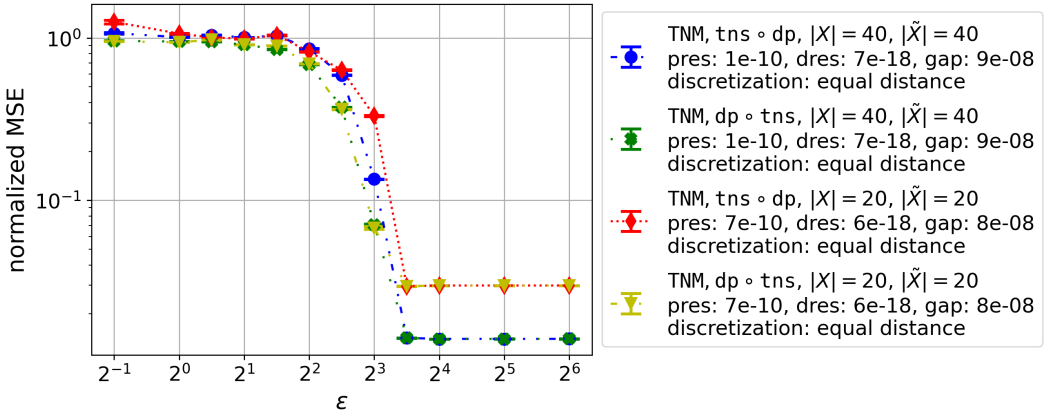


Figure 4.8: Comparison between a finer discretization and a coarser discretization of domains of sensitive features on ds2a (upon equal distance discretization)

Figure 4.9 illustrates a comparison between a finer discretization and a coarser discretization of domains of sensitive features on ds2a upon equal frequency discretization. We observe that a finer discretization of sensitive features results in an improvement in utility for both Model `tns` ∘ `dp` and Model `dp` ∘ `tns`. However, we remark that 40 bins were not enough for equal frequency discretization to result in gradual improvement of utility for a higher privacy budget.

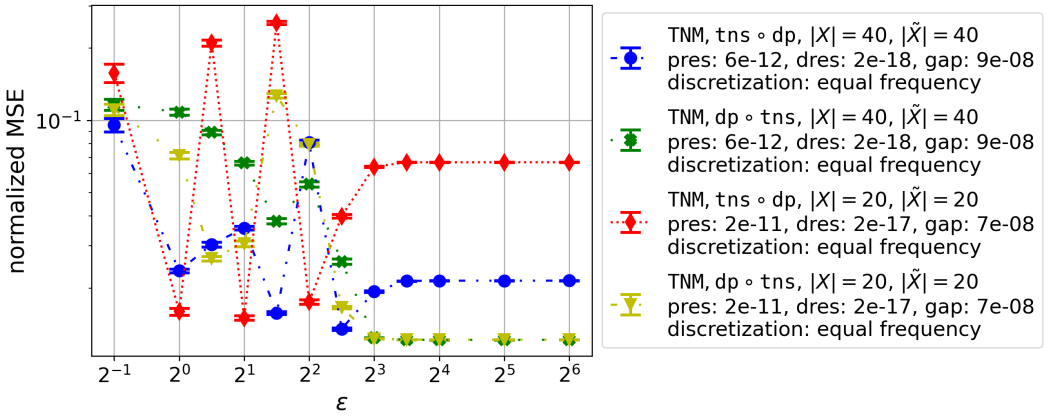


Figure 4.9: Comparison between a finer discretization and a coarser discretization of domains of sensitive features on ds2a (upon equal frequency discretization)

Figure 4.10 illustrates a comparison between a finer discretization and a coarser discretization of domains of privatized features on ds2a upon equal distance discretization, when there is one additional bin between every two bins of sensitive features (interpolation) and there are two additional bins one of which is lower and the other is higher than any other bin of sensitive features (extrapolation).

We observe that a finer discretization of privatized features does not result in a utility improvement upon a higher privacy budget. This is because, due the combination of the objective function for minimizing variance (Equation 4.21) and the differential privacy constraint (Equation 4.8), the obtained privatization functions have probabilities of 0 for sampling interpolated or extrapolated values.

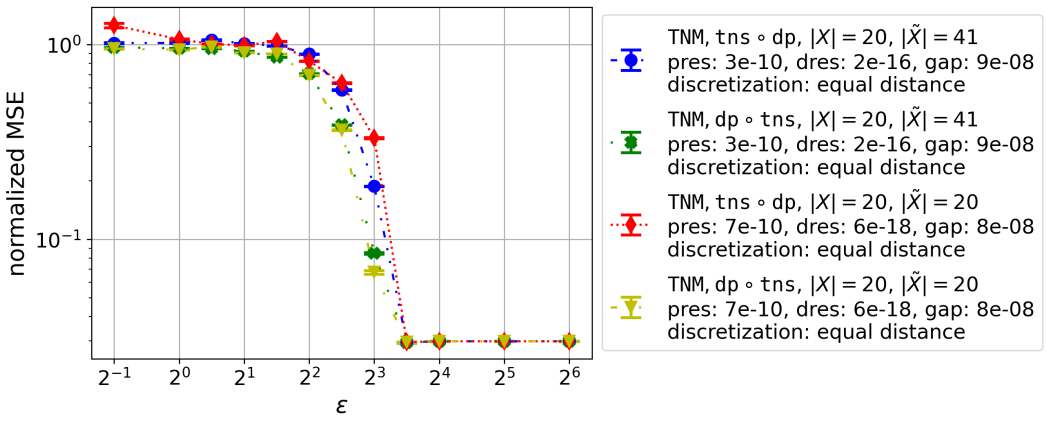


Figure 4.10: Comparison between a finer discretization and a coarser discretization of domains of privatized features on ds2a (upon equal distance discretization)

We remark that, in our experiments, the utility sometimes stagnates upon a higher privacy budget, e.g., this is the case in Figure 4.2. The cause of this is the regularization of the diagonal of the matrix $\frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X}$ in Equation 2.12, which mitigates the numerical difficulty of taking its matrix inverse (Equation 4.26) in exchange for lower accuracy.

## 4.7 Conclusion

We have provided a utility-maximizing mechanism for privatizing features that are computed by transforming sensitive attributes, when the transformations have high-magnitude gradients or singularities. In particular, we have provided the tailored noise mechanism (`TNM`) for privatizing features by solving a convex program in such a way that (i) only informative intervals of transformations are selected, (ii) the variance of privatization noise is minimized, and (iii) the biasedness of privatization noise is minimized.

In our experiments results on synthetic datasets and a synthetically-extended real dataset, we have observed that `TNM` results, typically, in a lower MSE between true target values and predicted target values compared to the Laplace mechanism (Definition 7), when fitting a linear regression model by regularized least squares (Equation 2.12) that use U-statistics, and when the feature function is either the logarithm, the reciprocal, or the tangent.

Also, in our experiment results (illustrated in Figure 4.2), we have observed that, in `TNM` and for the linear regression model of the ds2a dataset, Model `dp` ∘ `tns` (where attributes are transformed into features first, and only then differential privacy noise is added to the features) performs better than Model `tns` ∘ `dp` (where

differential privacy noise is added to attributes first, and only then the attributes are transformed into features) because the reciprocal transformation in ds2a approaches its singularity relatively quickly, thus the risk of obtaining outlier features is higher.

**Future directions.** Firstly, we remark that our approach of solving a convex program for obtaining the conditional probabilities of privatization functions is appropriate only for discrete privatization functions. To extend our approach for continuous privatization functions, we could attempt to identify a family of privatization functions with shape parameters, so that the convex program would remain tractable and intend to identify optimal shape parameters as opposed to conditional probabilities.

Secondly, we could attempt to identify a discretization strategy dedicated for models where attributes are transformed to several distinct features, and where the discretization strategy takes into account the influence of a weighted combination of distinct transformations. This way, we could determine the locations of the bins, which are informative over all distinct transformations.

# Chapter 5

# Summary and future directions

In Chapter 1, we have stated our principal question, which goes as follows: *how could the agents of a communication network maximize the accuracy of collaboratively computed averages of locally privatized attributes?* To answer this question, we have discussed two contributions, respectively, in Section 3.3 and Chapter 4. In the first work, we have discussed distributed averaging on graphs with arbitrary degree sequences, when the computed averages are unbiased, when the degrees of the agents are sensitive, and when the agents interact without handshakes. In the second work, we have discussed a utility-maximizing mechanism for privatizing features that are computed by transforming sensitive attributes, when the transformations have high-magnitude gradients or singularities.

In Section 5.1, we summarize the contributions by enumerating their advantages and disadvantages. In Section 5.2, we discuss future directions.

## 5.1 Summary of contributions

In this section, we summarize the two aforementioned contributions by enumerating their advantages and disadvantages, where the first contribution is the work on distributed averaging on graphs with arbitrary degree sequences, which is discussed in Section 3.3, and the second contribution is the work on tailored privatization, which is discussed in Chapter 4. For the work on tailored privatization, we also enumerate some remarks.

We remark that, in Section 3.2, we have provided an example of distributed averaging on ER graphs. However, ER graphs are not as general as graphs with arbitrary degree sequences because, in expectation, every vertex of an ER random graph has the degree $np$, where $n$ is the order of the graph and $p$ is the probability of edge assignment between any two distinct vertices of the ER random graph.

In the work on distributed averaging on graphs with arbitrary degree sequences, we have proved an asymptotic guarantee (Theorem 5) for the MSE between the average of sensitive attributes and the average of locally privatized attributes computed by the bias-correcting gossip algorithm (Equation 3.9), when the graph of agents is modeled by the configuration model (Definition 4), when the degrees of agents are sensitive, and when the agents interact without handshakes.

We enumerate the advantages of the work on distributed averaging on graphs with arbitrary degree sequences:

1. Equation 3.44 and Equation 3.45 indicate that distributed averaging performed by the bias-correcting gossip algorithm (`BCGo`) can result in less privatization noise in comparison to the averaging function $f(\mathbf{x}) = \frac{1}{n}\sum_{i\in[n]} x_i$ (where $\mathbf{x} \in \mathbb{R}^n$), when we compute U-statistics (Definition 1) of degree 1 from locally privatized degrees raised to a power $k = 2$, and when the graph has low degrees (i.e., when the 2-nd raw moment $\mu_{d^2}$ and the $k$-th raw moment $\mu_{d^k}$ of the degrees in Equation 3.44 are low).

2. Handshake-free interaction is a characteristic of a communication model with a stronger self-management. As a consequence, information dissemination among the agents is more robust upon inactive agents.

We enumerate the disadvantages of the work on distributed averaging on graphs with arbitrary degree sequences:

1. Equation 3.45 indicates that distributed averaging of locally privatized attributes performed by `BCGo` requires to split the privacy budget in $m+2$ parts and centralized averaging of locally privatized attributes requires to split the privacy budget in $m$ parts, where $m$ is the number of sensitive attributes. This way, the use of `BCGo` requires to split the privacy budget in 2 additional parts.

2. The heuristic (Equation 3.35) for the expected value of the reciprocal of the square of a Gaussian random variable, to our knowledge, has no closed-form expression. This way, the approximation error in Equation 3.29 is left unquantified.

In the work on tailored privatization, we have provided a utility-maximizing mechanism for privatizing features that are computed by transforming sensitive attributes, when the transformations have high-magnitude gradients or singularities. In particular, we have provided the tailored noise mechanism (`TNM`) for privatizing features by solving a convex program in such a way that (i) only informative intervals of transformations are selected, (ii) the variance of privatization noise is minimized, and (iii) the biasedness of privatization noise is minimized.

We enumerate the advantages of the work on tailored privatization:

1. In our experiment results on synthetic datasets and a synthetically-extended real dataset, we have observed that `TNM` results, typically, in a lower MSE between true target values and predicted target values compared to the Laplace mechanism (Definition 7), when fitting a linear regression model by regularized least squares that use U-statistics (Definition 1), and when the feature function is either the logarithm, the reciprocal, or the tangent.

2. In our experiment results (illustrated in Figure 4.2), we have observed that, in `TNM` and for the linear regression model of the ds2a dataset, Model `dp` $\circ$ `tns` (where attributes are transformed into features first, and only then differential

privacy noise is added to the features) performs better than Model `tns` ∘ `dp` (where differential privacy noise is added to attributes first, and only then the attributes are transformed into features) because ds2a has the reciprocal transformation which approaches its singularity relatively quickly, and the risk of obtaining outlier features is higher.

3. In our experiment results (illustrated in Figure 4.4 and Figure 4.5), we have observed that, in `TNM` and for the linear regression models of the ds2b dataset and the misra1d dataset, where two features are computed from distinct transformations of one attribute, Model `tns` ∘ `dp` performs better than Model `dp` ∘ `tns` because, for Model `dp` ∘ `tns`, the privacy budget was split in 2 parts and, for Model `tns` ∘ `dp`, the privacy budget remained unsplit.

4. The strategy of transforming the non-linear regression model of the original misra1d dataset to a linear one (Equation 4.27) can be generalized to other non-linear datasets.

We enumerate the disadvantages of the work on tailored privatization:

1. A utility-maximizing mechanism obtained by solving a convex program requires discretization of domains of privatization functions, and the presence of granularity results in utility loss.

2. We have found only one relevant real dataset (misra1d) where features are obtained from a transformation with a singularity, while misra1d is an extremely small dataset.

We enumerate some remarks of the work on tailored privatization:

1. In our experiment results (illustrated in Figure 4.6), we have observed that equal frequency discretisation performs better than equal distance discretization upon a transformation that approaches its singularity relatively slowly (for the discretization granularity that was handled by our machine).

2. In our experiment results (illustrated in Figure 4.7), we have observed that equal distance discretisation performs better than equal frequency discretization upon a transformations that approaches its singularity relatively quickly (for the discretization granularity that was handled by our machine).

## 5.2 Future directions

In this section, we firstly summarize more straightforward future directions of the work on distributed averaging on graphs with arbitrary degree sequences, which is discussed in Section 3.3, and the work on tailored privatization, which is discussed in Chapter 4. Then, we discuss future direction in a broader sense by considering the settings that are close to (yet outside of) the scope of this dissertation, as illustrated in Figure 2.5.

We discuss future directions of the work on distributed averaging on graphs with arbitrary degree sequences:

1. By identifying a strategy to obtain the number $n$ of neighbors, `BCGo` could be used to compute the unbiased sample variance which is a U-statistic of degree 2 (indicated by Equation 3.50). This way, we conjecture that, by mathematical induction, it can be shown that `BCGo` can compute U-statistics of arbitrary degree.

2. We conjecture that the convergence rate of `BCGo` executed on graphs with arbitrary degree sequences could be obtained from the transition matrix of the simple gossip algorithm (Algorithm 1), as `BCGo` is a combination of two distinct executions of the simple gossip algorithm (`SiGo`). Boyd et al. [Boy+06] indicates that the convergence rate of a gossip algorithm can be determined from the second largest singular value of the transition matrix. In such case, the convergence rate can be obtained from the spectral gap, which is discussed in more detail in the slides by Iutzeler [Iut16a; Iut16b].

3. For privatizing the degrees of agents, we have used the Gaussian mechanism (Definition 6). However, the degrees are always non-negative, thus their domain is bounded from below. A more suitable privatization mechanism could be one that results in privatization noise with a discrete support, e.g., the discrete Gaussian mechanism [CKS20] or `TNM`.

We discuss future directions of the work on tailored privatization:

1. Our approach of solving a convex program for obtaining the conditional probabilities of privatization functions is appropriate only for discrete privatization functions. To extend our approach for continuous privatization functions, we could attempt to identify a family of privatization functions with shape parameters, so that the convex program would remain tractable and intend to identify optimal shape parameters as opposed to conditional probabilities.

2. We could attempt to identify a discretization strategy dedicated for models where attributes are transformed to several distinct features, and where the discretization strategy takes into account the influence of a weighted combination of distinct transformations. This way, we could determine the locations of the bins, which are informative over all distinct transformations.

**Broader future directions.** We elaborate on the future directions in a broader sense. In Figure 2.5, we have illustrated two settings that are close to (yet outside of) the scope of this dissertation. One of such settings is related to dynamic communication models, and the other is related to personalized statistical models.

We start by discussing future directions towards dynamic communication models.

Firstly, we remark that communication models with a notion of time can be interpreted as complex networks (discussed in Subsection 2.2.1). Then, we highlight that the evolution of a complex network can be represented by its equilibrium state (a graph $G$) or by a sequence of graphs $(G_k)_{k\geq 1}$ of length $k$, where each graph in the sequence represents a state upon a particular instance of time.

We envision a future direction where the representation of a complex network by a graph $G$ could have an advantage over the representation of a complex network by a sequence $(G_k)_{k\geq 1}$. For example, if a graph $G$ was generated by adding vertices (and their edges) one by one (an example of such generation is the Chinese restaurant process discussed in Subsection 2.2.3), it might suffice to identify emergent properties (e.g., the presence of a giant component discussed in Subsection 2.2.1) by taking at $G$ rather than taking every element in the sequence $(G_k)_{k\geq 1}$ that led to $G$. This way, the computation cost of the identification of emergent properties of a dynamic communication model would be kept low.

We proceed by discussing future directions towards personalized statistical models.

Firstly, we remind that the bias between the average of locally privatized attributes computed by `SiGo` and the true average of individual values is present when the graph of agents is non-regular (as mentioned in Chapter 3). As indicated by Equation 3.45, `BCGo` can result in privatization noise that is higher than the corrected bias, when the bias is low and due to the additional splits of the privacy budget in `BCGo`. This way, we conjecture that there exist graphs on which `SiGo` can outperform `BCGo`, especially when the average degree is low.

We envision a future direction where `SiGo` is executed separately in clusters of agents, where each cluster is formed of agents who have similar degrees, as in such case the aforementioned bias would get low. This way, the future direction would involve a search of a convenient approach for clustering agents that have similar degrees.

One possible approach could be an application of a clustering algorithm based on $\epsilon$-regular pairs defined as follows. Let $\epsilon \in (0, 1)$. Let $C, C' \subseteq V$ be pairwise disjoint subsets of vertices of a graph. The pair $(C, C')$ is $\epsilon$-regular if for all $X \subseteq C$ and $Y \subseteq C'$, such that $|X| \geq \epsilon|C|$ and $|Y| \geq \epsilon|C'|$, we have

$$|\mathrm{den}(X, Y) - \mathrm{den}(C, C')| \leq \epsilon,$$

where $\mathrm{den}(X, Y)$ and $\mathrm{den}(C, C')$ are defined in Equation 2.6. As proven by Szemerédi's regularity lemma [Sze75], there exist such graphs for which it is possible to find a partition of vertices, where the partition includes pairwise disjoint subsets of vertices, and where each pair of subsets is $\epsilon$-regular. However, as indicated by Gowers [Gow97] and Moshkovitz and Shapira [MS16], such graphs are usually too large for tractability. We remark that the stochastic block model (Definition 3) is a convenient choice for modeling $\epsilon$-regular pairs because it is parametrized by the probabilities for edge assignments between every two communities.

We conjecture that a variant of a simple clustering algorithm (e.g., $k$-means clustering) could be designed in such a way that $\epsilon$-regularity would be used as an index and lead to almost $\epsilon$-regular pairs. If `SiGo` propagated only among vertices that belong to distinct sets of (almost) $\epsilon$-regular pairs, the propagation would happen among vertices that have similar degrees. This way, we would reduce the bias between the average of individual values computed by `SiGo` and the true average of individual values, when the graph is non-regular.

# Bibliography


[ADV20] Martin Andersen, Joachim Dahl, and Lieven Vandenberghe. "CVXOPT: Convex Optimization". In: *Astrophysics Source Code Library* (2020), ascl–2008. URL: https://ascl.net/2008.017.

[AEH75] E.A. Akkoyunlu, K. Ekanandham, and R.V. Huber. "Some Constraints and Tradeoffs in the Design of Network Communications". In: *Proceedings of the Fifth Symposium on Operating System Principles, SOSP 1975, The University of Texas at Austin, Austin, Texas, USA, November 19-21, 1975*. Ed. by James C. Browne and Juan Rodriguez-Rosell. ACM, 1975, pp. 67–74. DOI: 10.1145/800213.806523. URL: https://doi.org/10.1145/800213.806523.

[Alo+00] Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. "Efficient Testing of Large Graphs". In: *Comb.* 20.4 (2000), pp. 451–476. DOI: 10.1007/s004930070001. URL: https://doi.org/10.1007/s004930070001.

[AX18] Inês Almeida and João Xavier. "DJAM: Distributed Jacobi Asynchronous Method for Learning Personal Models". In: *IEEE Signal Process. Lett.* 25.9 (2018), pp. 1389–1392. DOI: 10.1109/LSP.2018.2859596. URL: https://doi.org/10.1109/LSP.2018.2859596.

[AY08] Charu C. Aggarwal and Philip S. Yu. "On static and dynamic methods for condensation-based privacy-preserving data mining". In: *ACM Trans. Database Syst.* 33.1 (2008), 2:1–2:39. DOI: 10.1145/1331904.1331906. URL: https://doi.org/10.1145/1331904.1331906.

[Ays+09] Tuncer C. Aysal, Mehmet E. Yildiz, Anand D. Sarwate, and Anna Scaglione. "Broadcast gossip algorithms for consensus". In: *IEEE Trans. Signal Process.* 57.7 (2009), pp. 2748–2761. DOI: 10.1109/TSP.2009.2016247. URL: https://doi.org/10.1109/TSP.2009.2016247.

[BA99] Albert-László Barabási and Réka Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (Oct. 1999), pp. 509–512. ISSN: 1095-9203. DOI: 10.1126/science.286.5439.509. URL: http://dx.doi.org/10.1126/science.286.5439.509.

[Bel+20] James Bell, Aurélien Bellet, Adrià Gascón, and Tejas Kulkarni. "Private Protocols for U-Statistics in the Local Model and Beyond". In: *AISTATS 2020 - 23rd International Conference on Artificial Intelligence and Statistics*. Palermo, Italy, Aug. 2020. URL: https://hal.inria.fr/hal-02310236.

[BGH20] Aurélien Bellet, Rachid Guerraoui, and Hadrien Hendrikx. "Who Started This Rumor? Quantifying the Natural Differential Privacy of Gossip Protocols". In: LIPIcs 179 (2020). Ed. by Hagit Attiya, 8:1–8:18. DOI: 10.4230/LIPIcs.DISC.2020.8. URL: https://doi.org/10.4230/LIPIcs.DISC.2020.8.

[BM16] Anirban Banerjee and Ranjit Mehatari. "An eigenvalue localization theorem for stochastic matrices and its application to Randić matrices". In: *Linear Algebra and its Applications* 505 (2016), pp. 85–96.

[BN10] Hai Brenner and Kobbi Nissim. "Impossibility of Differentially Private Universally Optimal Mechanisms". In: *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. IEEE Computer Society, 2010, pp. 71–80. DOI: 10.1109/FOCS.2010.13. URL: https://doi.org/10.1109/FOCS.2010.13.

[Boc+06] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and Dong-Uk Hwang. "Complex networks: Structure and dynamics". In: *Physics reports* 424.4-5 (2006), pp. 175–308.

[Bol84] Béla Bollobás. "The evolution of random graphs". In: *Transactions of the American Mathematical Society* 286.1 (1984), pp. 257–274.

[Boy+06] Stephen P. Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. "Randomized gossip algorithms". In: *IEEE Trans. Inf. Theory* 52.6 (2006), pp. 2508–2530. DOI: 10.1109/TIT.2006.874516. URL: https://doi.org/10.1109/TIT.2006.874516.

[BV14] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014. ISBN: 978-0-521-83378-3. DOI: 10.1017/CBO9780511804441. URL: https://web.stanford.edu/%5C%7Eboyd/cvxbook/.

[BW18] Borja Balle and Yu-Xiang Wang. "Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 394–403.

[CKS17] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. "Constrained Differential Privacy for Count Data". In: *CoRR* abs/1710.00608 (2017). arXiv: 1710.00608. URL: http://arxiv.org/abs/1710.00608.

[CKS20] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. "The Discrete Gaussian for Differential Privacy". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/b53b3a3d6ab90ce0268229151c9bde11-Abstract.html.

[CKS21] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. "Constrained Private Mechanisms for Count Data". In: *IEEE Trans. Knowl. Data Eng.* 33.2 (2021), pp. 415–430. DOI: 10.1109/TKDE.2019.2912179. URL: https://doi.org/10.1109/TKDE.2019.2912179.

[CL02] Fan Chung and Linyuan Lu. "Connected components in random graphs with given expected degree sequences". In: *Annals of combinatorics* 6.2 (2002), pp. 125–145.

[CLP11] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. "Rumor spreading in social networks". In: *Theor. Comput. Sci.* 412.24 (2011), pp. 2602–2610. DOI: 10.1016/j.tcs.2010.11.001. URL: https://doi.org/10.1016/j.tcs.2010.11.001.

[Col+16] Igor Colin, Aurélien Bellet, Joseph Salmon, and Stéphan Clémençon. "Gossip Dual Averaging for Decentralized Optimization of Pairwise Functions". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. 2016, pp. 1388–1396. URL: http://proceedings.mlr.press/v48/colin16.html.

[Cor+09] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms, third edition*. Computer science. MIT Press, 2009. ISBN: 9780262033848. URL: https://books.google.fr/books?id=i-bUBQAAQBAJ.

[CT15] C. Canuto and A. Tabacco. *Mathematical Analysis I*. Vol. 84. Springer International Publishing, 2015. ISBN: 9783319127729. URL: https://books.google.fr/books?id=wtr5BwAAQBAJ.

[DBR18] Pierre Dellenbach, Aurélien Bellet, and Jan Ramon. "Hiding in the Crowd: A Massively Distributed Algorithm for Private Averaging with Malicious Adversaries". In: *CoRR* abs/1803.09984 (2018). arXiv: 1803.09984. URL: http://arxiv.org/abs/1803.09984.

[DeG74] Morris H. DeGroot. "Reaching a consensus". In: *Journal of the American Statistical Association* 69.345 (1974), pp. 118–121.

[Dem+87] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. "Epidemic Algorithms for Replicated Database Maintenance". In: *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia,*

*Canada, August 10-12, 1987*. Ed. by Fred B. Schneider. ACM, 1987, pp. 1–12. DOI: 10.1145/41840.41841. URL: https://doi.org/10.1145/41840.41841.

[Dev86] L. Devroye. "Non-Uniform Random Variate Generation". In: Springer New York, 1986. ISBN: 9783540963059. URL: https://books.google.fr/books?id=mEw%5C_AQAAIAAJ.

[Die05] R. Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer Berlin Heidelberg, 2005. ISBN: 9783540261827. URL: https://books.google.fr/books?id=uTw0zgEACAAJ.

[DR14] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". In: *Found. Trends Theor. Comput. Sci.* 9.3-4 (2014), pp. 211–407. DOI: 10.1561/0400000042. URL: https://doi.org/10.1561/0400000042.

[DS12] M.H. DeGroot and M.J. Schervish. *Probability and Statistics*. Addison-Wesley, 2012. ISBN: 9780321500465. URL: https://books.google.fr/books?id=4TlEPgAACAAJ.

[DS81] Dorothy E. Denning and Giovanni Maria Sacco. "Timestamps in key distribution protocols". In: *Communications of the ACM* 24.8 (1981), pp. 533–536.

[Dwo+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. "Our Data, Ourselves: Privacy Via Distributed Noise Generation". In: *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*. 2006, pp. 486–503. DOI: 10.1007/11761679_29. URL: https://doi.org/10.1007/11761679%5C_29.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM, 2014, pp. 1054–1067. DOI: 10.1145/2660267.2660348. URL: https://doi.org/10.1145/2660267.2660348.

[ER59] P. Erdős and A. Rényi. "On Random Graphs I". In: *Publicationes Mathematicae Debrecen* 6 (1959), p. 290.

[ES10] B.S. Everitt and A. Skrondal. *The Cambridge Dictionary of Statistics*. Cambridge University Press, 2010. ISBN: 9780521766999. URL: https://books.google.fr/books?id=C98wSQAACAAJ.

[Fer03] Thomas S. Ferguson. "U-statistics". In: *Notes for Statistics* (2003).

[FH17] Lorenzo Federico and Remco van der Hofstad. "Critical Window for Connectivity in the Configuration Model". In: *Comb. Probab. Comput.* 26.5 (2017), pp. 660–680. DOI: 10.1017/S0963548317000177. URL: https://doi.org/10.1017/S0963548317000177.

[FK16] Alan Frieze and Michał Karoński. *Introduction to random graphs*. Cambridge University Press, 2016.

[GAM19] Simson L. Garfinkel, John M. Abowd, and Christian Martindale. "Understanding database reconstruction attacks on public data". In: *Commun. ACM* 62.3 (2019), pp. 46–53. DOI: 10.1145/3287287. URL: https://doi.org/10.1145/3287287.

[Gao+20] Pu Gao, Remco van der Hofstad, Angus Southwell, and Clara Stegehuis. "Counting Triangles in Power-Law Uniform Random Graphs". In: *Electron. J. Comb.* 27.3 (2020), p. 3. DOI: 10.37236/9239. URL: https://doi.org/10.37236/9239.

[Gas+17] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. "Privacy-Preserving Distributed Linear Regression on High-Dimensional Data". In: *PoPETs* 2017.4 (2017), pp. 345–364. DOI: 10.1515/popets-2017-0053. URL: https://doi.org/10.1515/popets-2017-0053.

[GG19] Lodovico Giaretta and Sarunas Girdzijauskas. "Gossip Learning: Off the Beaten Path". In: *2019 IEEE International Conference on Big Data (IEEE BigData), Los Angeles, CA, USA, December 9-12, 2019*. Ed. by Chaitanya K. Baru, Jun Huan, Latifur Khan, Xiaohua Hu, Ronay Ak, Yuanyuan Tian, Roger S. Barga, Carlo Zaniolo, Kisung Lee, and Yanfang (Fanny) Ye. IEEE, 2019, pp. 1117–1124. DOI: 10.1109/BigData47090.2019.9006216. URL: https://doi.org/10.1109/BigData47090.2019.9006216.

[GHS83] Robert G. Gallager, Pierre A. Humblet, and Philip M. Spira. "A Distributed Algorithm for Minimum-Weight Spanning Trees". In: *ACM Trans. Program. Lang. Syst.* 5.1 (1983), pp. 66–77. DOI: 10.1145/357195.357200. URL: https://doi.org/10.1145/357195.357200.

[Gia11] George Giakkoupis. "Tight bounds for rumor spreading in graphs of a given conductance". In: *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*. 2011, pp. 57–68. DOI: 10.4230/LIPIcs.STACS.2011.57. URL: https://doi.org/10.4230/LIPIcs.STACS.2011.57.

[GK11] Hoshin Vijai Gupta and Harald Kling. "On typical range, sensitivity, and normalization of Mean Squared Error and Nash-Sutcliffe Efficiency type metrics". In: *Water Resources Research* 47.10 (2011).

[GK16] Justin Gilmer and Swastik Kopparty. "A local central limit theorem for triangles in a random graph". In: *Random Struct. Algorithms* 48.4 (2016), pp. 732–750. DOI: 10.1002/rsa.20604. URL: https://doi.org/10.1002/rsa.20604.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004. ISBN: 0-521-83084-2. DOI: 10.1017/CBO9780511721656. URL: http://www.wisdom.weizmann.ac.il/%5C%7Eoded/foc-vol2.html.

[Gow97] W.T. Gowers. "Lower bounds of tower type for Szemerédi's uniformity lemma". In: *Geometric & Functional Analysis GAFA* 7.2 (May 1997), pp. 322–337. ISSN: 1420-8970. DOI: 10.1007/PL00001621. URL: https://doi.org/10.1007/PL00001621.

[GR11] Fan Chung Graham and Mary Radcliffe. "On the Spectra of General Random Graphs". In: *Electron. J. Comb.* 18.1 (2011). DOI: 10.37236/702. URL: https://doi.org/10.37236/702.

[GRS12] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. "Universally Utility-maximizing Privacy Mechanisms". In: *SIAM J. Comput.* 41.6 (2012), pp. 1673–1693. DOI: 10.1137/09076828X. URL: https://doi.org/10.1137/09076828X.

[GS10] Mangesh Gupte and Mukund Sundararajan. "Universally optimal privacy mechanisms for minimax agents". In: *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*. Ed. by Jan Paredaens and Dirk Van Gucht. ACM, 2010, pp. 135–146. DOI: 10.1145/1807085.1807105. URL: https://doi.org/10.1145/1807085.1807105.

[GS16] Göknur Giner and Gordon K. Smyth. "statmod: Probability Calculations for the Inverse Gaussian Distribution". In: *R J.* 8.1 (2016), p. 339. DOI: 10.32614/rj-2016-024. URL: https://doi.org/10.32614/rj-2016-024.

[GV16] Quan Geng and Pramod Viswanath. "The Optimal Noise-Adding Mechanism in Differential Privacy". In: *IEEE Trans. Inf. Theory* 62.2 (2016), pp. 925–951. DOI: 10.1109/TIT.2015.2504967. URL: https://doi.org/10.1109/TIT.2015.2504967.

[Hal60] P.R. Halmos. *Naive Set Theory*. Undergraduate texts in mathematics. Van Nostrand, 1960. ISBN: 9783540900924. URL: https://books.google.fr/books?id=-e1LAAAAMAAJ.

[Has70] W.K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1 (Apr. 1970), pp. 97–109. ISSN: 0006-3444. DOI: 10.1093/biomet/57.1.97. eprint: https://academic.oup.com/biomet/article-pdf/57/1/97/23940249/57-1-97.pdf. URL: https://doi.org/10.1093/biomet/57.1.97.

[Hay+09] Michael Hay, Chao Li, Gerome Miklau, and David D. Jensen. "Accurate Estimation of the Degree Distribution of Private Networks". In: *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*. Ed. by Wei Wang, Hillol Kargupta, Sanjay Ranka, Philip S. Yu, and Xindong Wu. IEEE Computer Society, 2009, pp. 169–178. DOI: 10.1109/ICDM.2009.11. URL: https://doi.org/10.1109/ICDM.2009.11.

[Hay+10] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. "Boosting the Accuracy of Differentially Private Histograms Through Consistency". In: *Proc. VLDB Endow.* 3.1 (2010), pp. 1021–1032. DOI: 10.14778/1920841.1920970. URL: http://www.vldb.org/pvldb/vldb2010/pvldb%5C_vol3/R91.pdf.

[HC91] Ian Hodkinson and Margaret Cunninghamas. *Computability, Algorithms, and Complexity*. 1991.

[HJ12] Roger A. Horn and Charles R. Johnson. *Matrix Analysis, 2nd Ed.* Cambridge University Press, 2012. ISBN: 9780521548236. DOI: 10.1017/CBO9781139020411. URL: https://doi.org/10.1017/CBO9781139020411.

[HLL83] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. "Stochastic blockmodels: First steps". In: *Social networks* 5.2 (1983), pp. 109–137.

[HMD14] Xi He, Ashwin Machanavajjhala, and Bolin Ding. "Blowfish privacy: tuning privacy-utility trade-offs using policies". In: *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*. Ed. by Curtis E. Dyreson, Feifei Li, and M. Tamer Özsu. ACM, 2014, pp. 1447–1458. DOI: 10.1145/2588555.2588581. URL: https://doi.org/10.1145/2588555.2588581.

[Hoe48] Wassily Hoeffding. "A Class of Statistics with Asymptotically Normal Distribution". In: *The Annals of Mathematical Statistics* 19.3 (1948), pp. 293–325. DOI: 10.1214/aoms/1177730196. URL: https://doi.org/10.1214/aoms/1177730196.

[Hoe63] Wassily Hoeffding. "Probability Inequalities for Sums of Bounded Random Variables". In: *Journal of the American Statistical Association* 58.301 (1963), pp. 13–30.

[Hof16] Remco van der Hofstad. *Random Graphs and Complex Networks*. Vol. 43. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2016. ISBN: 9781316779422. DOI: 10.1017/9781316779422. URL: https://doi.org/10.1017/9781316779422.

[Hof22] Remco van der Hofstad. *Random Graphs and Complex Networks*. Vol. 2. 2022. URL: https://www.win.tue.nl/~rhofstad/NotesRGCNII_colleagues_25_04_2022.pdf.

[Hsu+14] Justin Hsu, Marco Gaboardi, Andreas Haeberlen, Sanjeev Khanna, Arjun Narayan, Benjamin C. Pierce, and Aaron Roth. "Differential Privacy: An Economic Method for Choosing Epsilon". In: *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*. IEEE Computer Society, 2014, pp. 398–410. DOI: 10.1109/CSF.2014.35. URL: https://doi.org/10.1109/CSF.2014.35.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009. ISBN: 9780387848570. DOI: 10.1007/978-0-387-84858-7. URL: https://doi.org/10.1007/978-0-387-84858-7.

[Hua+21] Yifan Hua, Kevin Miller, Andrea L. Bertozzi, Chen Qian, and Bao Wang. "Efficient and Reliable Overlay Networks for Decentralized Federated Learning". In: *CoRR* abs/2112.15486 (2021). arXiv: 2112.15486. URL: https://arxiv.org/abs/2112.15486.

[ICH13] Franck Iutzeler, Philippe Ciblat, and Walid Hachem. "Analysis of Sum-Weight-Like Algorithms for Averaging in Wireless Sensor Networks". In: *IEEE Trans. Signal Process.* 61.11 (2013), pp. 2802–2814. DOI: 10.1109/TSP.2013.2256904. URL: https://doi.org/10.1109/TSP.2013.2256904.

[IMC21] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. "Locally Differentially Private Analysis of Graph Statistics". In: *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. Ed. by Michael Bailey and Rachel Greenstadt. USENIX Association, 2021, pp. 983–1000. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/imola.

[Iut16a] Franck Iutzeler. *Gossip algorithms: Tutorial and Recent advances (part 1)*. 2016. URL: https://www.iutzeler.org/assets/presentations/smile_iutzeler_part1.pdf.

[Iut16b] Franck Iutzeler. *Gossip algorithms: Tutorial and Recent advances (part 2)*. 2016. URL: https://www.iutzeler.org/assets/presentations/smile_iutzeler_part2.pdf.

[Jel11] Márk Jelasity. "Gossip". In: *Self-organising Software - From Natural to Artificial Adaptation*. Ed. by Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. Natural Computing Series. Springer, 2011, pp. 139–162. DOI: 10.1007/978-3-642-17348-6_7. URL: https://doi.org/10.1007/978-3-642-17348-6%5C_7.

[JMB09] Márk Jelasity, Alberto Montresor, and Özalp Babaoglu. "T-Man: Gossip-based fast overlay topology construction". In: *Computer Networks* 53.13 (2009), pp. 2321–2339. DOI: 10.1016/j.comnet.2009.03.013. URL: https://doi.org/10.1016/j.comnet.2009.03.013.

[JRA02] Lee W. Johnson, R. Dean Riess, and Jimmy T. Arnold. *Introduction to Linear Algebra*. Addison-Wesley, 2002. ISBN: 9780321190437.

[Kas+11] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. "What Can We Learn Privately?" In: *SIAM J. Comput.* 40.3 (2011), pp. 793–826. DOI: 10.1137/090756090. URL: https://doi.org/10.1137/090756090.

[Kas+13] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. "Analyzing Graphs with Node Differential Privacy". In: *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*. Ed. by Amit Sahai. Vol. 7785. Lecture Notes in Computer Science. Springer, 2013, pp. 457–476. DOI: 10.1007/978-3-642-36594-2_26. URL: https://doi.org/10.1007/978-3-642-36594-2%5C_26.

[KDG03] David Kempe, Alin Dobra, and Johannes Gehrke. "Gossip-Based Computation of Aggregate Information". In: *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*. 2003, pp. 482–491. DOI: 10.1109/SFCS.2003.1238221. URL: https://doi.org/10.1109/SFCS.2003.1238221.

[KK20] Martin Kenyeres and Jozef Kenyeres. "Applicability of Generalized Metropolis-Hastings Algorithm to Estimating Aggregate Functions in Wireless Sensor Networks". In: *Advances in Science, Technology and Engineering Systems Journal* 5.5 (2020), pp. 224–236. DOI: 10.25046/aj050528.

[KKM90] Ephraim Korach, Shay Kutten, and Shlomo Moran. "A Modular Technique for the Design of Efficient Distributed Leader Finding Algorithms". In: *ACM Trans. Program. Lang. Syst.* 12.1 (1990), pp. 84–101. DOI: 10.1145/77606.77610. URL: https://doi.org/10.1145/77606.77610.

[KS07] Anne-Marie Kermarrec and Maarten van Steen. "Gossiping in distributed systems". In: *Operating Systems Review* 41.5 (2007), pp. 2–7. DOI: 10.1145/1317379.1317381. URL: https://doi.org/10.1145/1317379.1317381.

[Lam+21] Antoine Lamer, Alexandre Filiot, Yannick Bouillard, Paul Mangold, Paul Andrey, and Jessica Schiro. "Specifications for the Routine Implementation of Federated Learning in Hospitals Networks". In: *Public Health and Informatics - Proceedings of MIE 2021, Medical Informatics Europe, Virtual Event, May 29-31, 2021*. Ed. by John Mantas, Lacramioara Stoicu-Tivadar, Catherine E. Chronaki, Arie Hasman, Patrick Weber, Parisis Gallos, Mihaela Marcella Vida, Emmanouil Zoulias, and Oana Sorina Chirila. Vol. 281. Studies in Health Technology and Informatics. IOS Press, 2021, pp. 128–132. DOI: 10.3233/SHTI210134. URL: https://doi.org/10.3233/SHTI210134.

[Les+08] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. "Statistical properties of community structure in large social and information networks". In: *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*. Ed. by Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang. ACM, 2008, pp. 695–704. DOI: 10.1145/1367497.1367591. URL: https://doi.org/10.1145/1367497.1367591.

[LK14] Jure Leskovec and Andrej Krevl. *SNAP Datasets: Stanford Large Network Dataset Collection*. `http://snap.stanford.edu/data`. June 2014.

[LMZ18] Gábor Lugosi, Shahar Mendelson, and Nikita Zhivotovskiy. "Concentration of the spectral norm of Erdős-Rényi random graphs". In: *arXiv e-prints* (2018). DOI: `10.48550/ARXIV.1801.02157`. URL: `https://arxiv.org/abs/1801.02157`.

[Lov93] László Lovász. "Random walks on graphs". In: *Combinatorics, Paul Erdős is eighty* 2.1-46 (1993), p. 4.

[LP09] Yehuda Lindell and Benny Pinkas. "Secure Multiparty Computation for Privacy-Preserving Data Mining". In: *J. Priv. Confidentiality* 1.1 (2009). DOI: `10.29012/jpc.v1i1.566`. URL: `https://doi.org/10.29012/jpc.v1i1.566`.

[LPW17] D.A. Levin, Y. Peres, and E.L. Wilmer. *Markov Chains and Mixing Times*. MBK. American Mathematical Society, 2017. ISBN: 9781470429621. URL: `https://books.google.fr/books?id=f208DwAAQBAJ`.

[McC02] Peter McCullagh. "What is a statistical model?" In: *The Annals of Statistics* 30.5 (2002), pp. 1225–1310. DOI: `10.1214/aos/1035844977`. URL: `https://doi.org/10.1214/aos/1035844977`.

[McM+17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*. Ed. by Aarti Singh and Xiaojin (Jerry) Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1273–1282. URL: `http://proceedings.mlr.press/v54/mcmahan17a.html`.

[Mil67] Stanley Milgram. "The Small-World Problem". In: *Psychology Today* 1.1 (1967), pp. 61–67.

[Mir12] Ilya Mironov. "On significance of the least significant bits for differential privacy". In: *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*. Ed. by Ting Yu, George Danezis, and Virgil D. Gligor. ACM, 2012, pp. 650–661. DOI: `10.1145/2382196.2382264`. URL: `https://doi.org/10.1145/2382196.2382264`.

[Mit97] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. ISBN: 978-0-07-042807-2. URL: `https://www.worldcat.org/oclc/61321007`.

[MM15] Connie Snyder Mick and Geoffrey Middlebrook. "Asynchronous and synchronous modalities". In: *Foundational practices of online writing instruction* (2015), pp. 129–148.

[MR96] Rajeev Motwani and Prabhakar Raghavan. "Randomized Algorithms". In: *ACM Comput. Surv.* 28.1 (1996), pp. 33–37. DOI: 10.1145/234313.234327. URL: https://doi.org/10.1145/234313.234327.

[MRT18] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning, second edition*. Adaptive Computation and Machine Learning series. MIT Press, 2018. ISBN: 9780262039406. URL: https://books.google.fr/books?id=V2B9DwAAQBAJ.

[MS16] Guy Moshkovitz and Asaf Shapira. "A short proof of Gowers' lower bound for the regularity lemma". In: *Combinatorica* 36.2 (Apr. 2016), pp. 187–194. ISSN: 1439-6912. DOI: 10.1007/s00493-014-3166-4. URL: https://doi.org/10.1007/s00493-014-3166-4.

[Nea+19] Joseph P. Near et al. "Duet: an expressive higher-order language and linear type system for statically enforcing differential privacy". In: *Proc. ACM Program. Lang.* 3.OOPSLA (2019), 172:1–172:30. DOI: 10.1145/3360598. URL: https://doi.org/10.1145/3360598.

[New03] Mark E.J. Newman. "The Structure and Function of Complex Networks". In: *SIAM Rev.* 45.2 (2003), pp. 167–256. DOI: 10.1137/S0036144450342480. URL: https://doi.org/10.1137/S003614450342480.

[New05] Mark E.J. Newman. "Power laws, Pareto distributions and Zipf's law". In: *Contemporary physics* 46.5 (2005), pp. 323–351.

[New10] Mark E.J. Newman. *Networks: An Introduction*. Oxford University Press, 2010. ISBN: 978-0-19920665-0. DOI: 10.1093/ACPROF:OSO/9780199206650.001.0001. URL: https://doi.org/10.1093/ACPROF:OSO/9780199206650.001.0001.

[Ngu19] Benjamin Nguyen. "Tutoriel: Anonymization Techniques: Theory and Practice". In: *Ecole d'été EGC*. 2019. URL: http://benjamin-nguyen.fr/EGC/Anonymisation-EGC-FINAL.pptx.

[NH21] Joseph P. Near and Xi He. "Differential Privacy for Databases". In: *Found. Trends Databases* 11.2 (2021), pp. 109–225. DOI: 10.1561/1900000066. URL: https://doi.org/10.1561/1900000066.

[Nik+13] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. "Privacy-Preserving Ridge Regression on Hundreds of Millions of Records". In: *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*. IEEE Computer Society, 2013, pp. 334–348. DOI: 10.1109/SP.2013.30. URL: https://doi.org/10.1109/SP.2013.30.

[NTC17] Erfan Nozari, Pavankumar Tallapragada, and Jorge Cortés. "Differentially private average consensus: Obstructions, trade-offs, and optimal algorithm design". In: *Autom.* 81 (2017), pp. 221–231. DOI: 10.1016/j.automatica.2017.03.016. URL: https://doi.org/10.1016/j.automatica.2017.03.016.

[Oli09] Roberto Imbuzeiro Oliveira. "Concentration of the adjacency matrix and of the Laplacian in random graphs with independent edges". In: *arXiv preprint arXiv:0911.0600* (2009).

[Paz63] A. Paz. "Graph-theoretic and algebraic characterizations of some Markov processes". In: *Israel Journal of Mathematics* 1.3 (1963), pp. 169–180.

[PCM22] Wei Peng, Tim Coleman, and Lucas Mentch. "Rates of convergence for random forests via generalized U-statistics". In: *Electronic Journal of Statistics* 16.1 (2022), pp. 232–292.

[Pen08] Chien-Yu Peng. "The first negative moment in the sense of the Cauchy principal value". In: *Statistics & Probability Letters* 78.13 (2008), pp. 1765–1774.

[Rad15] N.M. Radziwill. *Statistics with R: (the Easier Way)*. Lapis Lucera, 2015. ISBN: 9780692339428. URL: https://books.google.fr/books?id=QALGrQEACAAJ.

[RCP20] Marco Romanelli, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. "Optimal Obfuscation Mechanisms via Machine Learning". In: *33rd IEEE Computer Security Foundations Symposium, CSF 2020, Boston, MA, USA, June 22-26, 2020*. IEEE, 2020, pp. 153–168. DOI: 10.1109/CSF49147.2020.00019. URL: https://doi.org/10.1109/CSF49147.2020.00019.

[RFL19] Israel Donato Ridgley, Randy A. Freeman, and Kevin M. Lynch. "Simple, Private, and Accurate Distributed Averaging". In: *57th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2019, Monticello, IL, USA, September 24-27, 2019*. IEEE, 2019, pp. 446–452. DOI: 10.1109/ALLERTON.2019.8919736. URL: https://doi.org/10.1109/ALLERTON.2019.8919736.

[Ros16] David Rosenberg. *Excess Risk Decomposition*. 2016. URL: https://www.coursehero.com/file/13869964/2aexcess-risk-decomposition/.

[SB14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014. ISBN: 978-1-10-705713-5. URL: http://www.cambridge.org/de/academic/subjects/computer-science/pattern-recognition-and-machine-learning/understanding-machine-learning-theory-algorithms.

[SBR22] César Sabater, Aurélien Bellet, and Jan Ramon. "An accurate, scalable and verifiable protocol for federated differentially private averaging". In: *Mach. Learn.* 111.11 (2022), pp. 4249–4293. DOI: 10.1007/s10994-022-06267-9. URL: https://doi.org/10.1007/s10994-022-06267-9.

[Sha03] J. Shao. *Mathematical Statistics*. Springer Texts in Statistics. Springer, 2003. ISBN: 9780387953823. URL: https://books.google.fr/books?id=cyqTPotl7QcC.

[Sho15] Reza Shokri. "Privacy Games: Optimal User-Centric Data Obfuscation". In: *Proc. Priv. Enhancing Technol.* 2015.2 (2015), pp. 299–315. DOI: 10.1515/popets-2015-0024. URL: https://doi.org/10.1515/popets-2015-0024.

[SHS19] Daniel Silvestre, João Pedro Hespanha, and Carlos Silvestre. "Broadcast and Gossip Stochastic Average Consensus Algorithms in Directed Topologies". In: *IEEE Trans. Control of Network Systems* 6.2 (2019), pp. 474–486. DOI: 10.1109/TCNS.2018.2839341. URL: https://doi.org/10.1109/TCNS.2018.2839341.

[Sit+14] Ramesh K. Sitaraman, Mangesh Kasbekar, Woody Lichtenstein, and Manish Jain. "Overlay networks: An akamai perspective". In: *Advanced Content Delivery, Streaming, and Cloud Services* 51.4 (2014), pp. 305–328.

[SO22] Yuichi Sei and Akihiko Ohsuga. "Private True Data Mining: Differential Privacy Featuring Errors to Manage Internet-of-Things Data". In: *IEEE Access* 10 (2022), pp. 8738–8757. DOI: 10.1109/ACCESS.2022.3143813. URL: https://doi.org/10.1109/ACCESS.2022.3143813.

[Sot+21] Ekanut Sotthiwat, Liangli Zhen, Zengxiang Li, and Chi Zhang. "Partially Encrypted Multi-Party Computation for Federated Learning". In: *21st IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2021, Melbourne, Australia, May 10-13, 2021*. Ed. by Laurent Lefèvre, Stacy Patterson, Young Choon Lee, Haiying Shen, Shashikant Ilager, Mohammad Goudarzi, Adel Nadjaran Toosi, and Rajkumar Buyya. IEEE, 2021, pp. 828–835. DOI: 10.1109/CCGrid51090.2021.00101. URL: https://doi.org/10.1109/CCGrid51090.2021.00101.

[ST17] M. van Steen and A.S. Tanenbaum. *Distributed Systems*. CreateSpace Independent Publishing Platform, 2017. ISBN: 9781543057386. URL: https://books.google.fr/books?id=c77GAQAACAAJ.

[Ste10] M. van Steen. *Graph Theory and Complex Networks: An Introduction*. Maarten van Steen, 2010. ISBN: 9789081540612. URL: https://books.google.fr/books?id=V7bMbwAACAAJ.

[Sze75] E. Szemerédi. "On sets of integers containing k elements in arithmetic progression". eng. In: *Acta Arithmetica* 27.1 (1975), pp. 199–245. URL: http://eudml.org/doc/205339.

[Tel00] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 2000. ISBN: 9780521794831. URL: https://books.google.fr/books?id=vlpnS25qAJQC.

[TEN22] Sara Taki, Cédric Eichler, and Benjamin Nguyen. "It's Too Noisy in Here: Using Projection to Improve Differential Privacy on RDF Graphs". In: *ADBIS 2022-26th European Conference on Advances in Databases and Information Systems*. Vol. 1652. Springer International Publishing. 2022, pp. 212–221.

[Ter+22] Jean Ogier Du Terrail et al. "FLamby: Datasets and Benchmarks for Cross-Silo Federated Learning in Realistic Healthcare Settings". In: *NeurIPS 2022 - Thirty-sixth Conference on Neural Information Processing Systems*. Proceedings of NeurIPS. New Orleans, United States, Nov. 2022. URL: https://hal.science/hal-03900026.

[Tij07] H. Tijms. *Understanding Probability: Chance Rules in Everyday Life*. Cambridge University Press, 2007. ISBN: 9781139465458. URL: https://books.google.fr/books?id=Ua-%5C_5Ga4QF8C.

[Tsi84] John N. Tsitsiklis. "Problems in decentralized decision making and computation". PhD thesis. Massachusetts Institute of Technology, Cambridge, MA, USA, 1984. URL: http://hdl.handle.net/1721.1/15254.

[VBT17] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. "Decentralized Collaborative Learning of Personalized Models over Networks". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*. 2017, pp. 509–517. URL: http://proceedings.mlr.press/v54/vanhaesebrouck17a.html.

[Wan+21] Teng Wang, Jun Zhao, Zhi Hu, Xinyu Yang, Xuebin Ren, and Kwok-Yan Lam. "Local Differential Privacy for data collection and analysis". In: *Neurocomputing* 426 (2021), pp. 114–133. DOI: 10.1016/j.neucom.2020.09.073. URL: https://doi.org/10.1016/j.neucom.2020.09.073.

[Was04] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics. New York: Springer, 2004. ISBN: 978-1-4419-2322-6. DOI: 10.1007/978-0-387-21736-9.

[Wei12] N.A. Weiss. *Introductory Statistics*. Pearson Education, 2012. ISBN: 9780321691224. URL: https://books.google.fr/books?id=%5C_r5ucgAACAAJ.

[WS98] Duncan J. Watts and Steven H. Strogatz. "Collective dynamics of 'small-world' networks". In: *Nature* 393.6684 (1998), pp. 440–442. DOI: 10.1038/30918.

[WW13] Yue Wang and Xintao Wu. "Preserving Differential Privacy in Degree-Correlation based Graph Generation". In: *Trans. Data Priv.* 6.2 (2013), pp. 127–145. URL: http://www.tdp.cat/issues11/abs.a113a12.php.

[YS09] Xin Yan and Xiaogang Su. *Linear regression analysis: theory and computing*. World Scientific, 2009. DOI: 10.1142/6986. URL: https://www.worldscientific.com/doi/abs/10.1142/6986.

[ZC15] Mingyuan Zhou and Lawrence Carin. "Negative Binomial Process Count and Mixture Modeling". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 37.2 (2015), pp. 307–320. DOI: 10.1109/TPAMI.2013.211. URL: https://doi.org/10.1109/TPAMI.2013.211.

[ZC18] Alice Zheng and Amanda Casari. *Feature engineering for machine learning: principles and techniques for data scientists.*" O'Reilly Media, Inc.", 2018.

# Appendix A

# Averaging on arbitrary graphs

## A.1 Generation of graphs with power-law degree sequences

We describe a procedure for generating graphs with power-law degree sequences with the lowest degree $d_{\min} = 3$ and the highest degree $d_{\max} = 10^2$:

1. We generate a power-law degree sequence $\mathbf{d}'' = (d''_1, d''_2, \ldots, d''_n)$ whose elements $d''_1, d''_2, \ldots, d''_n$ are generated (drawn independently) from a probability distribution where a value $d''$ takes the following probability:

$$f_{\text{pow}}(d'' \,|\, \gamma) = \frac{(d'')^{-\gamma}}{\sum_{d \in [d_{\max}-3]} d^{-\gamma}},$$

   where $\gamma > 1$ and the support of the probability distribution is the set $\{1, 2, \ldots, d_{\max} - 3\}$. We remark that a power-law degree sequence is characterized by a higher proportion of vertices being attributed with lower degrees and a lower proportion of vertices being attributed with higher degrees.

2. We generate a graph $G'$ using the configuration model (Definition 4), where the configuration model is parametrized by the power-law degree sequence $\mathbf{d}''$. We denote the degree sequence of $G'$ by $\mathbf{d}'$.

3. For each $i \in [n]$, we check if $d'_i > d_{\max} - 3$ and if so we remove arbitrary edges that involve vertex $v_i$ until $d'_i = d_{\max} - 3$. Then, for each $i \in [n-1]$, we add the edge $\{v_i, v_{i+1}\}$; and, for $i = n$, we add the edge $\{v_n, v_1\}$ which guarantees that the graph is connected. If, for example, the edge $\{v_i, v_{i+1}\}$ was already present, we would try to add a subsequent edge that is absent, i.e., we would check the edges $\{v_i, v_{i+2}\}$, $\{v_i, v_{i+3}\}$, …, $\{v_i, v_{i-1}\}$. This guarantees that the degree of each vertex increases by 2. Further, if the edge $\{v_1, v_3\}$ is absent, we add it also because this guarantees that there is at least one cycle of odd length in the graph, as the presence of the edges $\{v_1, v_2\}$ and $\{v_2, v_3\}$ is assured by the aforementioned procedure that connects the graph. Finally, if the degree of some vertices is still lower than 3, we add some arbitrary edges so that the degree of each vertex is at least 3. We denote the resulting graph by $G$ and its degree sequence by $\mathbf{d}$.

Since the graph $G$ is connected and has at least one cycle of odd length, the principles of DeGroot learning indicate that the averages computed by Algorithm 1 involve the values of all agents in the communication network and that Algorithm 1 converges.

## A.2 Variance of a bounded attribute

We approximate the variance of a bounded (privatized) attribute by averaging the variances of truncated Gaussian random variables as follows:

$$(\tilde{\sigma}_k^{\mathrm{dp}})^2 \approx \frac{1}{n}\sum_{i\in[n]}(\sigma_k^{\mathrm{dp}})^2\left(1+\frac{\alpha_i\phi(\alpha_i)-\beta_i\phi(\beta_i)}{\omega(\beta_i)-\omega(\alpha_i)}-\left(\frac{\phi(\alpha_i)-\phi(\beta_i)}{\omega(\beta_i)-\omega(\alpha_i)}\right)^2\right),$$

where $\alpha_i = \frac{d_i^k - a_i - d_{i,k}^{\mathrm{dp}}}{\sigma_k^{\mathrm{dp}}}$, $\beta_i = \frac{d_i^k + a_i - d_{i,k}^{\mathrm{dp}}}{\sigma_k^{\mathrm{dp}}}$, $\phi$ is the probability density function of the standard Gaussian distribution, $\omega$ is the cumulative distribution of the standard Gaussian distribution, $\sigma_k^{\mathrm{dp}}$ is given in Equation 3.15, and $d_{i,k}^{\mathrm{dp}}$ is described in Remark 1. The approximation is due the summation of truncated Gaussian random variables. (The quantification of the approximation error is outside the scope of this work.)

## A.3 Remainder of Experiment 3.1

In Figure A.1, we illustrate the remaining result of Experiment 3.1 on the synthetic graph dataset. The interpretation of the result matches the case when $k = 1/2$, which is illustrated in Figure 3.1.

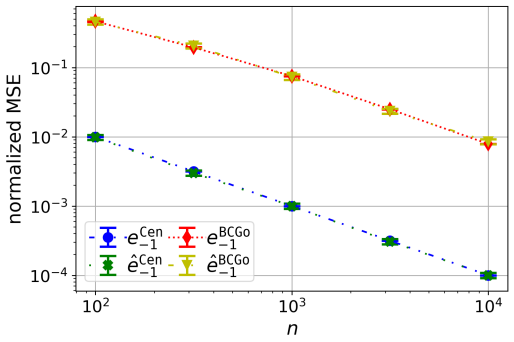


Figure A.1: Comparison of the expected errors (Equation 3.22 and Equation 3.29) and the empirical errors (Equation 3.47 and Equation 3.48), when $\epsilon = 2^2$. $k = -1$ refers to estimation of $\mu_{d^{-1}}$ (present in Equation 3.12).

## A.4 Secondary experiments

We have performed two sets of secondary experiments.

**Experiment A.1.** *Similarly as in Experiment 3.2, we compare the utility of the regression model fitted using the averages computed by* `BCGo`*, when degree sequences are generated, respectively, from the following shape parameters:* $\gamma \in \{2, 3, 4\}$.

**Experiment A.2.** *We evaluate the convergence of* `SiGo` *by comparing the variance of estimates of* $\mu_{d^{-1}}$ *over all agents, respectively, when the number of gossip iterations is the following:* $\mathrm{t_{go}} \in \{2^4, 2^{10}, 2^{11}\}$. *We consider the convergence sufficient if the variance is below* $10^{-10}$.

In Figure A.2 illustrates the results of Experiment A.1 on the synthetic graph dataset, which indicate that the MSE is lower when the shape parameter $\gamma$ is lower. For a higher $\gamma$, most of the vertices are of the lowest degree, and the variance due to privatization is low due to bounding indicated by Equation 3.17. Such case is more strongly susceptible to underfitting because the regression model is fitted mostly with features of lower variance.

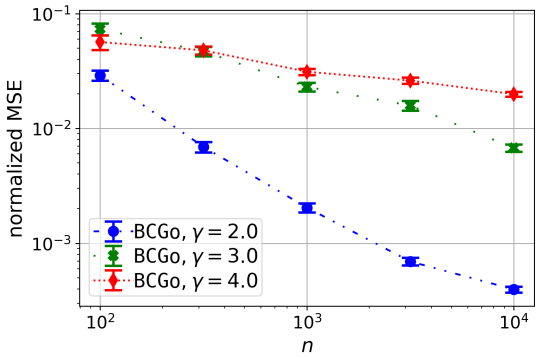


Figure A.2: Comparison of the MSE between true target values and predicted target values over several choices of the shape parameter $\gamma$

In Table A.1, the results of Experiment A.2 indicate that the variance of estimates of $\mu_{d^{-1}}$ (over all agents) gets significantly low when the number $\mathrm{t_{go}}$ of gossip iterations reaches $2^{10}$. This way, we conclude that $2^{10}$ are enough of gossip iterations for each of the evaluated datasets. The variance gets lower for the email network dataset sooner because its diameter is lower.

Table A.1: Empirical evaluation of the convergence of `SiGo`. We compare of the variance of estimates of $\mu_{d^{-1}}$ (over all agents) over several choices of the number $\mathrm{t_{go}}$ of gossip iterations

| $\mathrm{t_{go}}$ | email network | autonomous systems | synthetic graph ($n = 10^3$) |
|---|---|---|---|
| $2^4$ | $\approx 10^{-7}$ | $\approx 10^{-3}$ | $\approx 10^{-5}$ |
| $2^{10}$ | $\approx 10^{-33}$ | $\approx 10^{-17}$ | $\approx 10^{-33}$ |
| $2^{11}$ | $\approx 10^{-33}$ | $\approx 10^{-30}$ | $\approx 10^{-33}$ |

# Index