Locally Differentially Private Mechanisms for Sequential and High-Dimensional Data Analysis

by

Efehan Güner

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

January 2, 2025

Locally Differentially Private Mechanisms for Sequential and High-Dimensional Data Analysis

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Efehan Güner

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assist. Prof. Dr. Mehmet Emre Gürsoy (Advisor)

_____

Prof. Dr. Alptekin Küpçü

_____

Prof. Dr. Yücel Saygın

Date: _____

ABSTRACT

Locally Differentially Private Mechanisms for Sequential and High-Dimensional Data Analysis
Efehan Güner
Master of Science in Computer Science and Engineering
January 2, 2025

With the increasing need for data privacy and protection, local differential privacy (LDP) has emerged as a widely accepted standard for privacy-preserving data collection. While LDP has been studied extensively for singular data, its application to sequential and multidimensional data remains underexplored. In this thesis, we propose novel LDP mechanisms for advancing the state-of-the-art in the application of LDP to these two data types.

First, we propose PRIMA for learning discrete-time Markov chain models from sequential data under LDP. Markov chains are frequently used in the analysis and modeling of sequential data such as location traces, time series, natural language, and speech. However, considering that many such data sources are privacy-sensitive, it is imperative to design privacy-preserving methods for learning Markov chains. PRIMA addresses this need. In PRIMA, each user locally encodes and perturbs their sequential record on their own device using LDP protocols. For this purpose, we adapt two bitvector-based LDP protocols (RAPPOR and OUE); and furthermore, we develop a novel extension of the GRR protocol called AdaGRR. We also propose to utilize custom privacy budget allocation strategies for perturbation, which enable uneven splitting of the privacy budget to better preserve utility in cases with uneven sequence lengths. On the server side, PRIMA uses novel algorithms for estimating Markov probabilities from perturbed data. We experimentally evaluate PRIMA using three real-world datasets, four utility metrics, and various combinations of privacy budget and budget allocation strategies. Results show that PRIMA enables learning Markov chains with high utility and low error compared to Markov chains learned without privacy constraints.

Second, we propose MCM (Matrix-Based Data Collection Mechanism), a novel LDP mechanism for the collection of multidimensional data. In MCM, each user

encodes their multidimensional record into a bitmatrix. The rows of the bitmatrix are perturbed in a way that satisfies LDP. Then, the key contribution of MCM lies in its novel server-side estimation process, which enables the server to recover co-occurrence counts of all pairs of values in attribute domains. We utilize MCM to perform feature selection from multidimensional data using two popular feature selection metrics: information gain and chi-square. We experimentally evaluate MCM and the accompanying feature selection algorithms using three datasets, two utility metrics, and varying privacy budgets. Furthermore, we compare our solution with LDP-FS, a state-of-the-art solution for feature selection under LDP. We experimentally show that our solution yields more accurate information gain and chi-square values compared to LDP-FS, thereby improving the state-of-the-art. Furthermore, we demonstrate that correlations between attributes are accurately preserved in feature selection while LDP is satisfied.

ÖZETÇE

Sıralı ve Yüksek-Boyutlu Veri Analizi için Lokal Diferansiyel Mahremiyetli Mekanizmalar

Efehan Güner
Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans
2 Ocak 2025

Veri mahremiyeti ve koruması ihtiyacının artmasıyla birlikte, lokal diferansiyel mahremiyet (LDP), mahremiyet-korumalı veri toplanması için yaygın olarak kabul edilen bir standart haline gelmiştir. LDP tekil veriler için kapsamlı şekilde çalışılmış olsa da, LDP'nin sıralı ve yüksek-boyutlu verilere uygulanması hala yeterince araştırılmamıştır. Bu tezde, LDP'nin bu iki veri türüne uygulanmasındaki güncel literatürü ilerletmek için yeni LDP mekanizmaları öneriyoruz.

İlk olarak, sıralı verilerden LDP altında ayrık zamanlı Markov zinciri modellerinin öğrenilmesi için PRIMA'yı öneriyoruz. Markov zincirleri, konum izleri, zaman serileri, doğal dil ve konuşma gibi sıralı verilerin analizinde ve modellenmesinde sıkça kullanılmaktadır. Ancak, bu tür veri kaynaklarının çoğunun mahremiyet açısından hassas olduğu göz önüne alındığında, Markov zincirlerinin öğrenilmesi için mahremiyet korumalı yöntemler tasarlanması gerekmektedir. PRIMA, bu ihtiyacı karşılamaktadır. PRIMA'da, her kullanıcı sıralı veri kaydını kendi cihazında yerel olarak kodlar ve LDP protokollerini kullanarak bozar. Bu amaçla iki bitvektör tabanlı LDP protokolünü (RAPPOR ve OUE) uyarlıyoruz; ek olarak, GRR protokolünün yeni bir uzantısı olan AdaGRR'ı geliştiriyoruz. Ayrıca, dengesiz dizi uzunluklarına sahip durumlarda faydayı daha iyi korumak için mahremiyet bütçesinin eşit olmayan şekilde bölümünü sağlayan bütçe tahsis stratejileri kullanmayı öneriyoruz. Sunucu tarafında, PRIMA bozulmuş verilerden Markov olasılıklarını kestirmek için yeni algoritmalar kullanmaktadır. PRIMA'yı üç gerçek dünya veri kümesi, dört fayda ölçütü ve çeşitli mahremiyet bütçeleri ile bütçe tahsis stratejisi kombinasyonları kullanarak deneysel olarak değerlendiriyoruz. Sonuçlar, mahremiyet kısıtlamaları olmadan öğrenilen Markov zincirleri ile karşılaştırıldığında, PRIMA'nın yüksek fayda ve düşük hatalı Markov zincirleri öğrenmeyi sağladığını göstermektedir.

İkinci olarak, çok-boyutlu verilerin toplanması için yeni bir LDP mekanizması

olan MCM'yi (Matris-Tabanlı Veri Toplama Mekanizması) öneriyoruz. MCM'de, her kullanıcı çok-boyutlu kaydını bir bitmatrisi kullanarak kodlar. Bitmatrislerinin satırları LDP'yi sağlayacak şekilde bozulur. Ardından, MCM'nin temel katkısı, sunucunun öznitelik alanlarında yer alan tüm değer çiftlerinin eş oluşum sayılarını geri kazanmasını sağlayan yeni sunucu-taraflı kestirim sürecidir. MCM'yi, iki popüler öznitelik seçme metriği olan bilgi kazancı ve ki-kare kullanarak çok-boyutlu verilerden öznitelik seçimi yapmak amacıyla kullanıyoruz. MCM ve ona eşlik eden öznitelik seçme algoritmalarını üç veri kümesi, iki fayda ölçütü ve değişken mahremiyet bütçeleri kullanarak deneysel olarak değerlendiriyoruz. Ayrıca, çözümümüzü LDP altında öznitelik seçimi yapılmasını sağlayan en son yöntem olan LDP-FS ile karşılaştırıyoruz. Çözümümüzün deneysel olarak LDP-FS'e kıyasla daha doğru bilgi kazancı ve ki-kare değerleri sağladığını ve böylece literatürü ilerlettiğini gösteriyoruz. Ayrıca, LDP sağlanırken, öznitelik seçimi için öznitelikler arasındaki korelasyonların doğruluklu biçimde korunduğunu gösteriyoruz.

ACKNOWLEDGMENTS

First and foremost, I want to sincerely thank Assist. Prof. Dr. Mehmet Emre Gürsoy, my mentor and advisor, for his support and direction during my whole Master's program. Without him, I could not have developed as a researcher and finished my work.

I would like to thank Prof. Dr. Yücel Saygın from Sabancı University and Prof. Dr. Alptekin Küpçü from Koç University for serving on my thesis jury committee and for their insightful feedback and recommendations that helped to make my work better.

I would like to extend my thanks to my mother Azime Nurhan Güner and father Ali İhsan Güner for their love and support throughout my life. Additionally, I would like to thank all members of the SPADE Lab, especially Alireza Khodaie and Muhammed Esad Simitçioğlu for their support in location trajectory privacy research. I would like to thank my friend at Koç University, Arman Torikoğlu for his friendship and support during my MS studies. Special thanks to Ezgi Nur Mitil for her patience, inspiration and belief in me during this process.

Lastly, this research was supported by The Scientific and Technological Research Council of Turkiye (TUBITAK) under projects numbered 121E303 and 123E179. I sincerely thank TUBITAK for their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DP	Differential Privacy
LDP	Local Differential Privacy
OUE	Optimized Unary Encoding
MC	Markov Chain
IDE	Initial Distribution Error
TPE	Transition Probability Error
FPPE	First Passage Probability Error
IG	Information Gain
RMSE	Root Mean Squared Error

Chapter 1

INTRODUCTION

LDP has recently emerged as an accepted privacy standard and received significant attention from academia and the industry due to its effectiveness in protecting user data [Cormode et al., 2018a, Gursoy et al., 2022, Yang et al., 2020b]. LDP's ability to apply perturbation and achieve provable privacy before the data leaves the user's device has made it an appealing choice for academic research as well as companies aiming to incorporate strong privacy protection in their services. Consequently, LDP has been deployed in consumer-facing products of major tech companies such as Apple, Google, and Microsoft, marking a significant step towards enhancing user privacy in widely used digital products [app, 2020, Ding et al., 2017, Erlingsson et al., 2014].

In recent literature, LDP was commonly applied to singular data, i.e., each user has a single discrete or numeric true value [Murakami and Kawamoto, 2019, Gursoy et al., 2022, Gursoy et al., 2019, Wang et al., 2017, Wang et al., 2019a]. However, in the real world, sequential and multidimensional data are two fundamental data types that frequently exist in many contexts. Hence, there is a growing need to develop methods that apply LDP to these data types.

This thesis focuses on advancing the state-of-the-art in the application of LDP to sequential and multidimensional data. The thesis is composed of two main components: (i) PRIMA for learning discrete-time Markov chains from sequential data under LDP, and (ii) MCM for feature selection from multidimensional data under LDP. Next, we introduce these two components and state our main contributions in each component.

1.1 Introduction and Contributions of PRIMA

Sequential data arises naturally in a wide variety of contexts such as web browsing, location traces, time series, and natural language. A popular method for the analysis and modeling of sequential data is Markov chains [Gagniuc, 2017]. Markov chains are stochastic models that describe a sequence of possible events, where each event depends on one or more preceding events. Markov chain models have been successfully applied to sequential data such as location traces [Gambs et al., 2012, Cheng et al., 2016, Gursoy et al., 2018b, Wang et al., 2023a], natural language and speech recognition [Fine et al., 1998, Gales et al., 2008, Tokuda et al., 2013], as well as others [Meyn and Tweedie, 2012, Mor et al., 2020]. In recent years, Markov chain models have especially been used by the trajectory synthesis literature, considering that several systems internally rely on Markov chains, e.g., DPT [He et al., 2015], DP-Star [Gursoy et al., 2018a], AdaTrace [Gursoy et al., 2018b], PrivTrace [Wang et al., 2023a], and so forth [Gursoy et al., 2020, Wang and Kankanhalli, 2020].

On the other hand, sequential data used in learning a Markov chain model is oftentimes recorded and stored on users' devices, such as GPS locations, text messages and web browsing histories on users' smartphones. Furthermore, many of these data sources are privacy-sensitive, and the privacy of the data must be protected. Therefore, it is imperative to design privacy-preserving methods for learning Markov chain models.

In this thesis, we propose PRIMA for learning discrete-time Markov chain models under LDP. In PRIMA, each user locally encodes and perturbs their sequential record on their own device, using adapted versions of bitvector-based LDP protocols. This perturbation satisfies *sequence-level ε-LDP* such that the user's whole sequence is protected. To achieve LDP, we adapt and utilize two existing LDP protocols (RAPPOR and OUE) in PRIMA; in addition, we propose a novel extension of the GRR protocol called AdaGRR. As opposed to the original GRR protocol, our proposed AdaGRR protocol enables a bitvector-based implementation (which is consistent with RAPPOR and OUE) and can handle empty bitvectors which may arise in PRIMA.

In addition to the above, PRIMA contains two more novel design components: (i) budget allocation strategies for uneven splitting of the ε privacy budget, and (ii) a novel algorithm for server-side estimation of Markov transition probabilities. For the prior, considering that some users' sequences may be shorter and allocating uniform budget to each item may lead to wasted budgets in such cases, we propose decaying budget allocation strategies. Different budget allocation strategies are also experimentally shown to be preferable for maximizing different utility metrics. For the latter, a specialized algorithm is proposed in PRIMA for estimating Markov transition probabilities from LDP protocol outputs.

We experimentally evaluate PRIMA using three real-world sequential datasets, four utility metrics, four budget allocation strategies, and ε ranging between 0.5 and 5. Our utility metrics are designed to compare the LDP Markov chain model against the noise-free, non-private Markov chain model which would have been learned without privacy constraints. Overall, results show that PRIMA enables learning Markov chains with high utility and low error compared to their non-private counterparts. Furthermore, we observe that: (i) OUE and RAPPOR protocols perform similarly in terms of utility, (ii) AdaGRR has worse utility than OUE and RAPPOR for small ε but its utility converges to OUE and RAPPOR as ε is increased, (iii) despite its worse utility, AdaGRR is roughly twice as fast as RAPPOR and OUE, (iv) different budget allocation strategies may be preferred to maximize utility in terms of specific metrics, but the Step strategy performs well across the board for most utility metrics and ε values.

In summary, our main contributions are the following:

- We propose a new LDP protocol called AdaGRR, which is an extension of the GRR protocol. It can also be used in contexts outside of PRIMA and/or Markov chains.

- We design and develop PRIMA for learning Markov chains while satisfying sequence-level ε-LDP. A novel algorithm is proposed for server-side estimation of Markov transition probabilities from LDP protocol outputs.

- We propose budget allocation strategies for uneven splitting of the privacy budget and thereby improving the utility of Markov chain models.

- We perform an extensive experimental evaluation of PRIMA using three datasets, four utility metrics, varying privacy budgets, and budget allocation strategies.

1.2 Introduction and Contributions of MCM

The processing and analysis of multidimensional data, i.e., data where each record contains multiple attributes (features), has become critical in various domains, including machine learning, cybersecurity, census collection, finance, and healthcare analytics. Handling multidimensional data poses unique challenges due to the curse of dimensionality, which affects computational efficiency and model performance. As dimensionality increases, traditional algorithms often struggle with scalability, sparsity, and noise, necessitating advanced techniques such as dimensionality reduction, feature selection, and representation learning.

Consider the scenario where each user has a multidimensional record on their device. The server (data collector) would like to train a joint model or learn aggregate statistics from this data; however, due to the curse of dimensionality, the server wishes to perform *feature selection* beforehand to identify attributes that are of interest. Yet, due to privacy concerns, users would like to protect the privacy of their multidimensional records. Then, the following question arises: Can we perform feature selection while satisfying a provable privacy guarantee?

In this thesis, we propose methods to perform feature selection while satisfying LDP. Our methods are based on a novel mechanism that satisfies LDP, called "Matrix-Based Data Collection Mechanism" (MCM). In MCM, each user encodes their record into a bitmatrix. Then, the rows of the bitmatrix are perturbed one by one – we formally prove that this perturbation satisfies LDP. Then, the key contribution of MCM lies in the server-side estimation process. We derive and mathematically prove unbiased estimators to recover co-occurrence counts of value pairs, which are necessary for feature selection computations. We then utilize these co-occurrence counts to compute information gain (IG) and chi-square (χ^2), which

are two popular feature selection metrics, between pairs of attributes. We analyze all of our methods in terms of time and space complexity. We find that our user-side methods have linear complexities, and although the complexities of the server-side methods are larger (e.g., quadratic), we find that they can be executed on a commodity laptop within a few minutes.

We experimentally evaluate MCM and the accompanying feature selection methods using three datasets, two utility metrics, and ε ranging between 0.3 and 2.1. Furthermore, we compare our solution with LDP-FS [Alishahi et al., 2022], a state-of-the-art solution for feature selection under LDP. We observe that the Root Mean Squared Error (RMSE) of IGs and χ^2 computed using our solution are usually lower than LDP-FS, oftentimes substantially. This shows that our methods improve the state-of-the-art in LDP feature selection. In addition, we perform a heatmap analysis of IG and χ^2 values between all attributes in the datasets, to observe if the relative IG and χ^2 between attributes are preserved. We find that: (i) errors in noisy heatmaps decrease as ε is increased, and (ii) although the IG and χ^2 values may be noisy due to LDP, the trends regarding which two attributes are more correlated versus less correlated are preserved despite LDP. This shows that if the data collector were to rank attributes in terms of IG or χ^2 and select high-ranked ones (i.e., eliminate low-ranked ones), the data collector's selections (or eliminations) would be correct.

In summary, our main contributions are the following:

- We design and develop MCM for collecting multidimensional records under LDP. The key novelty and contribution of MCM lies in the derivation of server-side estimations of co-occurrence counts between all value pairs.

- Using MCM, we propose an end-to-end solution to compute IG and χ^2, which are two popular metrics for feature selection, between all pairs of attributes while satisfying LDP.

- We perform an experimental evaluation of our solution using varying datasets, metrics, and privacy budgets. We experimentally show that our solution per-

forms better than LDP-FS, the state-of-the-art solution for LDP feature selection [Alishahi et al., 2022].

1.3 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we survey and present the related work. In Chapter 3, we describe PRIMA and its algorithms. In Chapter 4, we perform an experimental evaluation of PRIMA. In Chapter 5, we describe MCM and the accompanying algorithms for feature selection under LDP. In Chapter 6, we perform an experimental evaluation of MCM. Finally, Chapter 7 concludes this thesis.

Chapter 2

RELATED WORK

2.1 Applications of LDP to Sequential Data

In recent years, LDP has emerged as an accepted privacy standard and received significant attention from the industry and academia [Cormode et al., 2018a, Ding et al., 2017, Erlingsson et al., 2014, Yang et al., 2020b]. While LDP has been applied to various kinds of data, related to PRIMA in particular are the applications of LDP to sequential data. Xiong et al. [Xiong et al., 2016] worked on sequential sensor data and proposed a randomized requantization scheme that satisfies LDP. Wang et al. [Wang et al., 2018a] proposed PrivTrie for frequent term discovery from text, utilizing a trie data structure to store counts of character sequences which are obtained using randomized response. In [Wang et al., 2019c], Wang et al. studied the problem of heavy hitter identification and proposed the PEM method which utilizes LDP protocols such as GRR and OLH. Ye et al. [Ye et al., 2021] studied the application of LDP to time series data. To preserve temporal order, they proposed an extended threshold-based mechanism. Errounda and Liu [Errounda and Liu, 2021] designed a sliding window-based approach for releasing collective location statistics under LDP. Cunningham et al. [Cunningham et al., 2021] studied the application of LDP to trajectory data. Their method perturbs trajectories in contiguous subsequences of length n called n-grams. Although all of these methods apply LDP to some form of sequential data, they differ from PRIMA since they do not learn Markov chains.

2.2 Learning Markov Chains under Centralized DP

While Markov chain learning has not been previously studied under LDP, Markov chains have been studied in centralized DP. Chen et al. [Chen et al., 2012] utilized

algorithms such as Laplace mechanism for learning Markov models and publishing sequential databases. Shen and Yu [Shen and Yu, 2013] studied the discovery of frequent patterns from graph databases under centralized DP. For frequent pattern mining, they used a Markov Chain Monte Carlo (MCMC) based sampling algorithm. Fan et al. [Fan et al., 2014] studied the analysis of web browsing data under centralized DP, and utilized Markov chains for multivariate time series modeling. Xiao et al. [Xiao et al., 2017] proposed a location cloaking system called LocLok, which uses Markov chain modeling to make inferences regarding users' true locations. Chen et al. [Chen et al., 2023] proposed word differential privacy, which is an extension of DP. They proposed mechanisms to satisfy this definition when collecting data from symbolic systems, and utilized Markov chains for sequence generation. In addition, Markov chain models were commonly used in the literature for DP trajectory (location trace) synthesis. Several tools developed for this purpose rely on Markov chain models, including DPT [He et al., 2015], DP-Star [Gursoy et al., 2018a], AdaTrace [Gursoy et al., 2018b], PrivTrace [Wang et al., 2023a], and others [Gursoy et al., 2020, Wang and Kankanhalli, 2020]. The key difference between PRIMA and the works surveyed in this section is that PRIMA satisfies LDP, whereas the works surveyed in this section satisfy centralized DP.

2.3 Applications of LDP to Multidimensional Data

Applications of LDP to multidimensional data are relevant to MCM since MCM works on multidimensional data. One line of work in this domain specializes in answering range queries on multidimensional data under LDP. In [Yang et al., 2020a], Yang et al. introduced HDG for answering multi-attribute range queries on tabular data by combining multiple low-dimensional grids to answer higher-dimensional range queries. In [Wang et al., 2019b], Wang et al. proposed HIO, aimed at handling multidimensional, SQL-like analytical queries on fact tables under LDP. In [Du et al., 2021a], Du et al. developed AHEAD, which employs an adaptive hierarchical domain decomposition approach that iteratively aligns with the underlying data distribution, demonstrating superior utility compared to both HIO and HDG

in empirical evaluations. In [Wang et al., 2023b], Wang et al. proposed PrivNUD in which the domain is subdivided into sub-domains of customized granularity, sub-domains with low density are discarded, and an adaptive user allocation method is used to estimate density within each remaining sub-domain. Furthermore, Wang and Cheng [Wang and Cheng, 2022] proposed PRISM to collect high-dimensional data under LDP by leveraging prefix-sum based data cubes. MCM differs from these works because although it works on multidimensional data, it does not aim to specialize in answering range queries.

Another line of work focuses on frequency estimation from multidimensional data. In [Arcolezi et al., 2021], Arcolezi et al. proposed an approach based on random sampling plus fake data for computing multidimensional frequency estimates with LDP. In [Arcolezi et al., 2024], Arcolezi et al. proposed methods for improving the utility of LDP protocols for longitudinal and multidimensional frequency estimates. In [da Costa Filho and Machado, 2023], the FELIP approach was developed for frequency estimation and multidimensional query answering on multidimensional datasets, which utilizes the concept of grids. MCM differs from these works since MCM's goal is not frequency estimation, though it can be used for frequency estimation as well.

A third line of work focuses on releasing (publishing) multidimensional datasets or marginals derived from multidimensional datasets under LDP. Ren et al. [Ren et al., 2018] proposed LoPub for high-dimensional crowdsourced data publication with LDP. Wang et al. [Wang et al., 2019d] proposed a locally private method based on copula functions for high-dimensional data release. Liu et al. [Liu et al., 2023] proposed an incremental learning-based probabilistic graphical model (PGM) construction method for multidimensional data publishing with LDP. Zhang et al. [Zhang et al., 2018] and Cormode et al. [Cormode et al., 2018b] studied the marginal release problem under LDP. MCM differs from these works since MCM's goal is not multidimensional dataset or marginal release.

There are several other works which apply LDP to multidimensional data. In [Wang et al., 2019a], Wang et al. studied the collection and analysis of multidimensional numeric data under LDP. As a case study, they applied their methods

to build an LDP-compliant stochastic gradient descent (SGD) algorithm, which is a building block for many machine learning models. Shen et al. [Shen et al., 2021] proposed PLDP, a personalized LDP definition, for multidimensional data aggregation. They developed a new LDP protocol called PMOUE to satisfy the PLDP definition. In [Du et al., 2021b], Du et al. studied the collection of correlation-constrained high-dimensional data collection under LDP. In [Qian et al., 2023], Qian et al. proposed collaborative sampling-based methods for partial collection of multidimensional records under LDP. Makhlouf et al. [Makhlouf et al., 2024] studied the fairness impacts of multidimensional data collection with LDP. Most closely related to MCM is LDP-FS developed by Alishahi et al. [Alishahi et al., 2022], which enables feature selection on multidimensional data under LDP. We experimentally compare MCM with LDP-FS and show that MCM generally provides higher utility than LDP-FS.

Chapter 3

PRIMA: LEARNING MARKOV CHAINS UNDER LDP

In this chapter, we describe PRIMA for learning discrete-time Markov chain models from sequential data under LDP.

3.1 Background and Preliminaries

3.1.1 Problem Setting and Notation

Consider a user population $\mathcal{P} = \{u_1, u_2, ..., u_{|\mathcal{P}|}\}$ and the finite domain of items $\mathcal{D} = \{I_1, I_2, ..., I_{|\mathcal{D}|}\}$. For each user $u \in \mathcal{P}$, the user's data is a sequence denoted by S_u. Formally, S_u is an ordered sequence of items $S_u = I_i \rightarrow I_j \rightarrow I_k \rightarrow ...$, such that $\forall I \in S_u$, it holds that: $I \in \mathcal{D}$. It is possible for an item to occur multiple times and/or consecutively in S_u. This formalism of sequential data is versatile and its semantic meaning can vary between applications. For example, in the context of web browsing, each item I corresponds to one webpage, S_u stores which pages u visited (ordered by time), and $\mathcal{D}$ corresponds to the set of reachable pages on the website. In the context of location traces, each item I corresponds to a location (can be discretized using grids or map-matching to roads) and S_u stores the user's time-ordered location trace.

In the rest of the chapter, the following notations are used: For a sequence S_u, $|S_u|$ denotes its length and $S_u[i]$ denotes the i'th element in the sequence, e.g., $S_u[1]$ is the first element of S_u. Same notations are used for 1-dimensional arrays (vectors). For 2-dimensional arrays (matrices), say matrix M, we denote by $M[i]$ the i'th row of the matrix and by $M[i][j]$ the element at row i, column j.

3.1.2 Local Differential Privacy (LDP)

Each user's sequence S_u is kept locally on the user's device. The data collector (server) would like to learn a Markov chain model from users' sequences, but users are unwilling to share their sequences with the data collector due to privacy concerns. In this scenario, *Local Differential Privacy (LDP)* enables privacy-preserving learning of a Markov chain. In LDP, each user's sequence is encoded and perturbed on their local devices in a way that satisfies LDP. Afterwards, the perturbed outputs are shared with the data collector. Since only perturbed outputs are observed by the data collector and not the true data, LDP can be used in scenarios where users do not trust the data collector. Due to its desirable trust assumption and provable privacy, LDP has also been used in consumer-facing applications of companies such as Apple, Google and Microsoft [Cormode et al., 2018a, Ding et al., 2017, Erlingsson et al., 2014].

When users' data is a sequence rather than a single item, LDP can be defined at two granularities: *item-level* or *sequence-level*. Sequence-level LDP aims to protect the user's entire sequence rather than a single item; thus, it is stronger than item-level LDP. Therefore, we adopt the sequence-level LDP definition which is formalized below.

Definition 3.1.1 (Sequence-level ε-LDP)**.** Let S_u, S'_u be two sequences such that $\forall I \in S_u$ and $\forall I' \in S'_u$, it holds that: $I \in \mathcal{D}$ and $I' \in \mathcal{D}$, where $\mathcal{D}$ denotes the domain. A randomized algorithm Ψ satisfies sequence-level ε-local differential privacy (ε-LDP), if and only if:

$$\forall y \in Range(\Psi): \quad \frac{\Pr[\Psi(S_u) = y]}{\Pr[\Psi(S'_u) = y]} \leq e^{\varepsilon} \tag{3.1}$$

where $\varepsilon > 0$ is the privacy parameter and $Range(\Psi)$ denotes the set of all possible outputs of Ψ.

ε-LDP ensures that given the perturbed output y, it is not possible for the data collector to distinguish whether the original true value was S_u or S'_u beyond the probability ratio controlled by e^{ε}. Here, ε is the privacy parameter (also known as the *privacy budget*). Lower ε yields stronger privacy.

ID	Items in Sequence
S_1	$I_1 \rightarrow I_2$
S_2	$I_2 \rightarrow I_3 \rightarrow I_1$
S_3	$I_1 \rightarrow I_1 \rightarrow I_2 \rightarrow I_4$
S_4	I_3
S_5	$I_4 \rightarrow I_3 \rightarrow I_2$
S_6	$I_3 \rightarrow I_3 \rightarrow I_3$

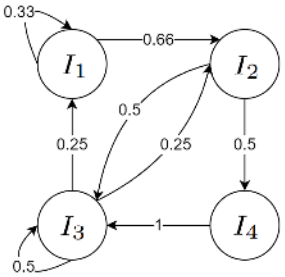

Figure 3.1: Sample data and Markov chain model

3.1.3 Markov Chains

A Markov chain is a stochastic model that describes a sequence of possible events, where each event depends on one or more preceding events. It is frequently used in the analysis and modeling of sequential data such as location traces, time series, natural language, speech, and others [Cheng et al., 2016, Gambs et al., 2012, Gursoy et al., 2018b, Gursoy et al., 2018a, Wang et al., 2023a, Gales et al., 2008, Tokuda et al., 2013, Meyn and Tweedie, 2012]. A Markov chain is said to be of k'th order if each event in a sequence depends on the k events preceding it, rather than all preceding events. For example, if each event depends only on one preceding event, then the order of the Markov chain is 1.

A Markov chain consists of two components: a set of states and a set of transition probabilities which represent the probabilities of moving from one state to another. In this thesis, the set of states corresponds to items (or combinations of items) from $\mathcal{D}$. Then, the Markov property can be formalized as follows.

Definition 3.1.2 (Markov property)**.** Given a set of discrete states $\mathcal{D}$, a sequence $S = I_1 \rightarrow I_2 \rightarrow ... \rightarrow I_n$ follows a k'th order Markov process if for all $I \in \mathcal{D}$ and $k \leq i \leq n-1$:

$$\Pr\big[S[i+1]\big|S[i]...S[1]\big] = \Pr\big[S[i+1]\big|S[i]...S[i-k+1]\big] \tag{3.2}$$

where $\Pr\big[S[i+1]\big|S[i]...S[i-k+1]\big]$ is called the transition probability from $S[i-k+1]...S[i]$ to $S[i+1]$.

In order to build a Markov chain model, transition probabilities need to be

learned from users' data. Let z denote a subsequence and zx denote a subsequence in which z is immediately followed by x. The transition probability $\Pr[I_j|z]$ is defined as:

$$\Pr[I_j|z] = \frac{\sum_{u \in \mathcal{P}} \varphi_u(zI_j)}{\sum_{u \in \mathcal{P}} \sum_{x \in \mathcal{D}} \varphi_u(zx)} \tag{3.3}$$

where $\varphi_u(zx)$ is a function that counts the number of occurrences of zx in user u's sequence S_u. A k'th order Markov chain contains transition probabilities $\Pr[I_j|z]$ for all $I_j \in \mathcal{D}$ and all subsequences z with length $|z| = k$. An example first order Markov chain ($k = 1$) is given in Figure 3.1. Since $\mathcal{D} = \{I_1, I_2, I_3, I_4\}$, there exist four states. The transition probabilities between states are computed according to the application of Equation 3.3 to the sequences given in the figure.

3.1.4 Problem Statement

The server would like to learn a k'th order Markov chain from users' sequences. If users' sequences could be collected without privacy concerns, it would be possible to learn the Markov chain by computing $\varphi_u(zx)$ for all combinations of z and x and thereby learning the transition probabilities via Equation 3.3. Let $\mathcal{M}$ denote this noise-free, gold-standard Markov chain model. However, when sequence-level ε-LDP needs to be satisfied, it is not possible to collect users' sequences in plaintext and learn $\mathcal{M}$. Instead, a *noisy* Markov chain model will be constructed (say $\mathcal{M}'$) using LDP. From a utility perspective, it is desired to minimize the difference (error) between $\mathcal{M}$ and $\mathcal{M}'$, which we denote by: $Err(\mathcal{M}, \mathcal{M}')$. Then, the problem that we study in this chapter is to propose algorithms such that the server learns a Markov chain model $\mathcal{M}'$ while satisfying sequence-level ε-LDP for each user $u \in \mathcal{P}$ and the model has small $Err(\mathcal{M}, \mathcal{M}')$.

3.2 LDP Protocols

Before explaining our solution (PRIMA) to the problem stated in Section 3.1.4, we describe the LDP protocols used in PRIMA. Several LDP protocols were developed in the literature for minimizing utility loss or communication cost under varying

conditions. In PRIMA, we adapt and use three LDP protocols: RAPPOR, OUE, and AdaGRR (adapted version of GRR). Among them, RAPPOR and OUE are from previous works. AdaGRR is our extension of the existing GRR protocol, and therefore a novel contribution of this thesis. Following convention, we introduce the protocols assuming each user's true value v_u is one item. We allow either $v_u \in \mathcal{D}$ or v_u is empty (i.e., $v_u = \emptyset$).

3.2.1 RAPPOR

RAPPOR was originally developed by Google and implemented in Chrome [Erlingsson et al., 2014]. Although the original version of RAPPOR uses Bloom filters to encode strings, variants of RAPPOR were deployed in later works for data encoded in a one-hot (unary) bitvector [Gursoy et al., 2022, Wang et al., 2017], which is used in PRIMA and explained below.

User u initializes bitvector B_u with length $|\mathcal{D}|$. The user sets $B_u[v_u] = 1$, and for all remaining positions $j \neq v_u$, $B_u[j] = 0$. Then, the perturbation step of RAPPOR takes as input B_u and outputs a perturbed bitvector B'_u. Perturbation algorithm Ψ_{RAP} considers each bit in B_u one-by-one, and either keeps or flips it with probability:

$$\forall_{i \in [1,|\mathcal{D}|]}: \quad \Pr[B'_u[i] = 1] = \begin{cases} \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1} & \text{if } B_u[i] = 1 \\ \frac{1}{e^{\varepsilon/2}+1} & \text{if } B_u[i] = 0 \end{cases} \tag{3.4}$$

The user sends B'_u to the server.

The server receives perturbed bitvectors B'_u from all users $u \in \mathcal{P}$. To perform estimation for value v, $Sup(v)$ is computed as the total number of received bitvectors that satisfy: $B'_u[v] = 1$. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{Sup(v) + |\mathcal{P}| \cdot (\rho - 1)}{2\rho - 1} \tag{3.5}$$

where ρ is the bit keeping probability: $\rho = \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}$.

3.2.2 Optimized Unary Encoding (OUE)

Optimized Unary Encoding (OUE) was originally proposed in [Wang et al., 2017]. OUE also uses one-hot bitvector encoding; however, it differs from RAPPOR in its asymmetric treatment of 0 and 1 bits. In OUE, if the original bit is 1, it is kept or flipped with equal probability $\frac{1}{2}$. If the original bit is 0, it is kept with high probability $\frac{e^\varepsilon}{e^\varepsilon+1}$ and flipped with low probability $\frac{1}{e^\varepsilon+1}$.

Similar to RAPPOR, user u initializes bitvector B_u with length $|\mathcal{D}|$, sets $B_u[v_u] = 1$, and for all remaining positions $j \neq v_u$, $B_u[j] = 0$. Then, the perturbation algorithm Ψ_{OUE} takes as input B_u and outputs perturbed bitvector B'_u such that:

$$\forall_{i\in[1,|\mathcal{D}|]}: \quad \Pr[B'_u[i]=1] = \begin{cases} \frac{1}{2} & \text{if } B_u[i]=1 \\ \frac{1}{e^\varepsilon+1} & \text{if } B_u[i]=0 \end{cases} \tag{3.6}$$

The user sends B'_u to the server.

The server receives perturbed bitvectors B'_u from all users $u \in \mathcal{P}$. To perform estimation for value v, $Sup(v)$ is computed as the total number of received bitvectors that satisfy: $B'_u[v] = 1$. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{2\cdot\left((e^\varepsilon+1)\cdot Sup(v) - |\mathcal{P}|\right)}{e^\varepsilon - 1} \tag{3.7}$$

3.2.3 Adapted GRR (AdaGRR)

Generalized Randomized Response (GRR) is a generalization of the randomized response survey technique [Cormode et al., 2018a, Wang et al., 2017, Wang et al., 2018b]. However, the original version of GRR is not suitable for Prima due to two reasons: (i) it does not use bitvector-based encoding, (ii) even if it is modified to use bitvector-based encoding, it cannot handle empty bitvectors (all 0s) which may arise in Prima. Hence, we develop a novel adaptation of GRR, which we call AdaGRR.

Let v_u be u's true item, and let $q = \frac{e^\varepsilon}{e^{2\varepsilon}+(|\mathcal{D}|-1)(e^\varepsilon-1)}$ and $p = \frac{q(e^\varepsilon-1)}{1-q}$ be AdaGRR protocol parameters. AdaGRR outputs perturbed bitvector $B'_u \in \{0,1\}^{|\mathcal{D}|}$ which is

computed as follows:

$$B'_u = \begin{cases} \{0\}^{|\mathcal{D}|} & \text{with probability } q \\ \Psi_{\text{AGRR}}(v_u) & \text{with probability } p(1-q) \\ \Psi_{\text{RAND}}(v_u) & \text{with probability } (1-p)(1-q) \end{cases} \tag{3.8}$$

$\Psi_{\text{RAND}}(v_u)$ generates a random one-hot bitvector B'_u such that $B'_u[v_u] \neq 1$. That is, B'_u will contain exactly one 1 bit, but this 1 bit must be at an index other than v_u. $\Psi_{\text{AGRR}}(v_u)$ behaves as follows: If $v_u \in \mathcal{D}$, then B'_u is initialized full of 0s, and then $B'_u[v_u] = 1$ is enforced. Otherwise, i.e., $v_u = \emptyset$, then B'_u is full of 0s. The user sends B'_u to the server.

The server receives perturbed bitvectors B'_u from all users $u \in \mathcal{P}$. To perform estimation for value v, $Sup(v)$ is computed as the total number of received bitvectors that satisfy: $B'_u[v] = 1$. Furthermore, $Sup(\emptyset)$ is computed as the total number of received bitvectors B'_u which consist only of 0 bits. Then, the server estimates $\bar{C}(\emptyset)$ as:

$$\bar{C}(\emptyset) = \frac{Sup(\emptyset) - q|\mathcal{P}|}{p(1-q)} \tag{3.9}$$

Finally, the server uses $\bar{C}(\emptyset)$ to compute the estimate $\bar{C}(v)$ for value v:

$$\bar{C}(v) = \frac{(1-q)(1-p)|\mathcal{P}| - \frac{\bar{C}(\emptyset)}{|\mathcal{D}|} - Sup(v)(|\mathcal{D}|-1)}{(1-q)(1-p|\mathcal{D}|)} \tag{3.10}$$

Since AdaGRR is a novel protocol proposed in this thesis, here we provide proofs that AdaGRR's user-side perturbation (Equation 3.8) satisfies LDP and AdaGRR's estimations are unbiased (Equations 3.9 and 3.10).

Theorem 1. *AdaGRR's user-side perturbation satisfies ε-LDP.*

Proof. Let Ψ denote the user-side perturbation process of AdaGRR, let v_u and v'_u denote two possible values of the user, and let B' denote the perturbed bitvector produced by AdaGRR. We need to show that:

$$\forall B' : \frac{\Pr[\Psi(v_u) = B']}{\Pr[\Psi(v'_u) = B']} \leq e^{\varepsilon} \tag{3.11}$$

Let P_{max} denote the maximum probability that the numerator can take and P_{min} denote the minimum probability that the denominator can take. That is:

$$P_{max} = \max(\Pr[\Psi(v_u) = B']) \tag{3.12}$$

$$P_{min} = \min(\Pr[\Psi(v'_u) = B']) \tag{3.13}$$

Then, in order to show that ε-LDP is satisfied, it is sufficient to prove that $\frac{P_{max}}{P_{min}} \leq e^{\varepsilon}$. Notice that there are two distinct cases for B' according to the definition of AdaGRR: (i) B' is all 0s or (ii) B' is a one-hot encoded bitvector (contains a single 1 bit). Below, we prove that $\frac{P_{max}}{P_{min}} \leq e^{\varepsilon}$ holds for both cases.

Case 1: B' is all 0s.

In this case, for P_{max} to be maximized, v_u should be equal to: $v_u = \emptyset$. For P_{min} to be minimized, v'_u should be a value from $\mathcal{D}$, i.e., $v_u \in \mathcal{D}$. According to these selections of v_u and v'_u and according to the user-side perturbation of AdaGRR, we compute:

$$\frac{P_{max}}{P_{min}} = \frac{(1-q)p + q}{q} \tag{3.14}$$

Since $p = \frac{q(e^{\varepsilon}-1)}{1-q}$, we have:

$$\frac{P_{max}}{P_{min}} = \frac{q(e^{\varepsilon}-1)+q}{q} = \frac{qe^{\varepsilon}-q+q}{q} = e^{\varepsilon} \tag{3.15}$$

Case 2: B' is a one-hot bitvector.

There are three distinct possibilities for B' to be obtained: (i) with probability $(1-q)p$ the original index of v_u is preserved, i.e., $B'[v_u] = 1$, (ii) with probability $\frac{(1-q)(1-p)}{|\mathcal{D}|}$ the user's true value was $v_u = \emptyset$, (iii) with probability $\frac{(1-q)(1-p)}{|\mathcal{D}|-1}$ the user's true value was $v_u \in \mathcal{D}$ but $B'[v_u] \neq 1$, i.e., a different index became 1. Among these three possibilities, taking into account the values of p and q, we observe that:

$$P_{max} = (1-q)p \tag{3.16}$$

$$P_{min} = \frac{(1-q)(1-p)}{|\mathcal{D}|-1} \tag{3.17}$$

Consequently, and by plugging in $p = \frac{q(e^{\varepsilon}-1)}{1-q}$ and $q = \frac{e^{\varepsilon}}{e^{2\varepsilon}+(|\mathcal{D}|-1)(e^{\varepsilon}-1)}$, we obtain:

$$\frac{P_{max}}{P_{min}} = \frac{p(|\mathcal{D}|-1)}{1-p} = \frac{q(e^{\varepsilon}-1)(|\mathcal{D}|-1)}{1-e^{\varepsilon}q} = \frac{\frac{e^{\varepsilon}(e^{\varepsilon}-1)(|\mathcal{D}|-1)}{(e^{\varepsilon}-1)(|\mathcal{D}|-1)+e^{2\varepsilon}}}{\frac{(e^{\varepsilon}-1)(|\mathcal{D}|-1)}{(e^{\varepsilon}-1)(|\mathcal{D}|-1)+e^{2\varepsilon}}} = e^{\varepsilon} \tag{3.18}$$

□

Theorem 2. *AdaGRR's server-side estimation of $\bar{C}(\emptyset)$ is unbiased.*

Proof. Let $C(\emptyset)$ denote the true number of users in the population for whom $v_u = \emptyset$. We need to show that $\mathbb{E}[\bar{C}(\emptyset)] = C(\emptyset)$.

$$\mathbb{E}[\bar{C}(\emptyset)] = \mathbb{E}\left[\frac{Sup(\emptyset) - q|\mathcal{P}|}{p(1-q)}\right] = \frac{\mathbb{E}[Sup(\emptyset)] - q|\mathcal{P}|}{p(1-q)} \tag{3.19}$$

Based on the user-side perturbation process of AdaGRR, we have:

$$\mathbb{E}[Sup(\emptyset)] = C(\emptyset)(q + p(1-q)) + (|\mathcal{P}| - C(\emptyset))q \tag{3.20}$$

Plugging Equation 3.20 into Equation 3.19, we get:

$$\mathbb{E}[\bar{C}(\emptyset)] = \frac{C(\emptyset)(q + p - pq) + q|\mathcal{P}| - qC(\emptyset) - q|\mathcal{P}|}{p(1-q)} \tag{3.21}$$

$$= \frac{C(\emptyset)(p - pq)}{p(1-q)} = \frac{C(\emptyset)p(1-q)}{p(1-q)} = C(\emptyset) \tag{3.22}$$

□

Theorem 3. *For $v \in \mathcal{D}$, AdaGRR's server-side estimation of $\bar{C}(v)$ is unbiased.*

Proof. Let $C(v)$ denote the true number of users in the population for whom $v_u = v$. We need to show that $\mathbb{E}[\bar{C}(v)] = C(v)$. We compute:

$$\mathbb{E}[\bar{C}(v)] = \mathbb{E}\left[\frac{(1-q)(1-p)|\mathcal{P}| - \frac{\bar{C}(\emptyset)}{|\mathcal{D}|} - Sup(v)(|\mathcal{D}| - 1)}{(1-q)(1-p|\mathcal{D}|)}\right] \tag{3.23}$$

$$= \frac{(1-p)|\mathcal{P}|}{1-p|\mathcal{D}|} - \frac{C(\emptyset)}{(1-q)(1-p|\mathcal{D}|)|\mathcal{D}|} - \frac{\mathbb{E}[Sup(v)](|\mathcal{D}| - 1)}{(1-q)(1-p|\mathcal{D}|)} \tag{3.24}$$

This is because $\mathbb{E}[\bar{C}(\emptyset)] = C(\emptyset)$ by Theorem 2. Then, based on the user-side perturbation process of AdaGRR, we have:

$$\mathbb{E}[Sup(v)] = C(v)(1-q)p + \frac{C(\emptyset)(1-q)(1-p)}{|\mathcal{D}|} + \frac{(|\mathcal{P}| - C(\emptyset) - C(v))(1-q)(1-p)}{|\mathcal{D}| - 1} \tag{3.25}$$

Plugging Equation 3.25 in place of $\mathbb{E}[Sup(v)]$ in Equation 3.24 and continuing with arithmetic operations, we reach:

$$\mathbb{E}[\bar{C}(v)] = \frac{C(v)(1-q)(1-p) - (|\mathcal{D}| - 1)C(v)(1-q)p}{(1-q)(1-p|\mathcal{D}|)} \tag{3.26}$$

$$= \frac{C(v)(1 - p - p|\mathcal{D}| + p)}{1-p|\mathcal{D}|} = C(v) \tag{3.27}$$

□

Algorithm 1: Overview of PRIMA

Input : $\mathcal{P}$, ε, $\mathcal{D}$, k, max length ϑ

Output: Markov chain $\mathcal{M}'$

`/* Server-side setup phase */`

1 $\langle \varepsilon_1, \varepsilon_2, ..., \varepsilon_\vartheta \rangle \leftarrow$ BUDGET_ALLOCATE(ε, ϑ)

2 Server sends $\langle \varepsilon_1, \varepsilon_2, ..., \varepsilon_\vartheta \rangle$ to all users $u \in \mathcal{P}$

3 $responses \leftarrow \{\}$

`/* Each user computes noisy LDP response (bitmatrix` M'_u`) and sends response to server */`

4 **foreach** $u \in \mathcal{P}$ **do**

5 Let S_u denote u's sequence; $|S_u| \leq \vartheta$

6 $M'_u \leftarrow$ ENCODE_PERTURB$(S_u, \mathcal{D}, \langle \varepsilon_1, \varepsilon_2, ..., \varepsilon_\vartheta \rangle)$

7 User sends M'_u to server

8 Server adds M'_u to $responses$

`/* Server learns Markov Chain */`

9 $\mathcal{M}' \leftarrow$ LEARN_MC$(\mathcal{D}, k, responses, \langle \varepsilon_1, \varepsilon_2, ..., \varepsilon_\vartheta \rangle)$

10 **return** $\mathcal{M}'$

3.3 Overview of our Solution: Prima

We propose PRIMA for solving the problem stated in Section 3.1.4. An overview of PRIMA is given in Algorithm 1. The inputs are the user population $\mathcal{P}$, privacy budget ε, domain $\mathcal{D}$, and a maximum length parameter ϑ. The ϑ parameter is used to enforce a maximum length limit on users' sequences. On line 1, Algorithm 1 uses the BUDGET_ALLOCATE function which divides the total ε budget into ϑ pieces, collectively denoted by $\langle \varepsilon_1, \varepsilon_2, ..., \varepsilon_\vartheta \rangle$. Details of the BUDGET_ALLOCATE function are given in Section 3.4. The server shares $\langle \varepsilon_1, \varepsilon_2, ..., \varepsilon_\vartheta \rangle$ with $\mathcal{P}$ and initializes an empty collection called *responses*, which will store the noisy LDP responses collected from users. Then, between lines 4-8, each $u \in \mathcal{P}$ encodes and perturbs his/her sequence S_u into a bitmatrix, and sends the resulting bitmatrix M'_u to the

server. Details of user-side encoding and perturbation are given in Section 3.5. Finally, having collected and stored users' perturbed matrices inside *responses*, the server uses $\mathcal{D}$, k and *responses* to learn the Markov chain on line 9. Details of Markov chain learning are given in Section 3.6.

3.4 Budget Allocation in Prima

The budget allocation step takes as input the total ε budget and divides it into ϑ pieces denoted by $\langle \varepsilon_1, \varepsilon_2, ..., \varepsilon_\vartheta \rangle$ such that: $\varepsilon_1 + \varepsilon_2 + ... + \varepsilon_\vartheta = \varepsilon$. Considering that S_u consists of up to ϑ items, budget allocation facilitates the assignment of ε_i to the perturbation of the i'th item of S_u. This summation ensures that the overall perturbation of the whole S_u satisfies sequence-level ε-LDP.

A straightforward way to divide ε into ϑ pieces is to do it evenly: $\varepsilon_1 = \varepsilon_2 = ... = \varepsilon_\vartheta = \frac{\varepsilon}{\vartheta}$, which we call *Uniform Allocation*. While this allocation strategy is legitimate, it does not provide much flexibility, and it is not always the best strategy utility-wise due to two main reasons. First, ϑ is an upper limit on sequence lengths, but some users' sequences may be much shorter than ϑ. In that case, budget allocated to perturb non-existing items will not be utilized effectively, causing redundant utility loss. Instead, by allocating higher budget to early items (e.g., ε_1, ε_2) and lower budget to later items (e.g., $\varepsilon_{\vartheta-1}$, ε_ϑ), we can decrease the amount of budget wasted. Second, certain budget allocations may be preferred to maximize certain utility metrics. For example, if the goal is to have a highly accurate initial state distribution for the Markov chain model, then higher budget can be allocated to the first items while lower budget can be allocated for the rest. In contrast, a more balanced budget allocation may be preferred if the goal is to preserve overall transition probabilities.

The different budget allocation strategies we consider in PRIMA are as follows:

1. **Uniform:** Budgets are allocated equally, i.e.: $\varepsilon_i = \frac{\varepsilon}{\vartheta}$ for all $i \in [1, \vartheta]$.

2. **Linear Decay:** Each subsequent budget is reduced linearly by a decay rate denoted by d. That is: $\varepsilon_i = \varepsilon_{i-1} - d$. The first budget ε_1 is chosen to satisfy

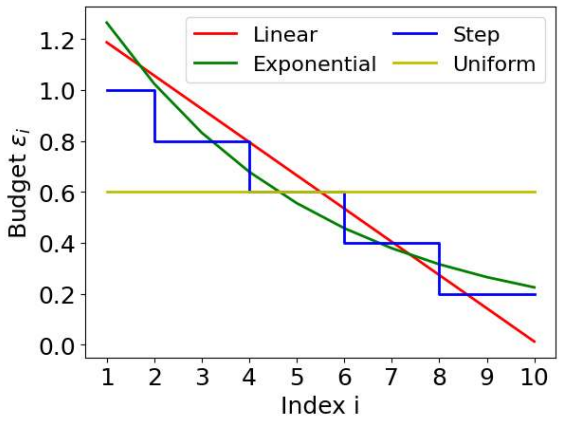

Figure 3.2: Comparison of budget allocation strategies ($\varepsilon = 6$, $\vartheta = 10$).

$\sum_{i=1}^{\vartheta} \varepsilon_i = \varepsilon$.

3. **Exponential Decay:** Each subsequent budget is reduced compared to the previous budget; however, as opposed to linear decay, the decay occurs using the probability mass function of a discrete Exponential distribution: $\varepsilon_i = (1-\gamma)^i \gamma\varepsilon$. Here, γ is the parameter that controls the decay rate.

4. **Step:** This strategy reduces the budget in several steps. Initially, ε_1 is highest. The next budgets $\varepsilon_2, ..., \varepsilon_\alpha$ are equal to ε_1. Then, $\varepsilon_{\alpha+1}$ is equal to $\varepsilon_\alpha - \beta$, i.e., after α equal budgets, a reduction of β is applied. Then, the next budgets $\varepsilon_{\alpha+2}, ..., \varepsilon_{2\alpha}$ are equal to $\varepsilon_{\alpha+1}$. Then, $\varepsilon_{2\alpha+1}$ is equal to $\varepsilon_{2\alpha} - \beta$, and so forth. In general, every α steps, a reduction of β is enforced.

A visual comparison of the different budget allocation strategies is given in Figure 3.2. In this figure, we use $d = 0.12$ for Linear Decay, $\gamma = 0.2$ for Exponential Decay, and $\alpha = 2$ and $\beta = 0.2$ for the Step strategy. The total ε is $\varepsilon = 6$, which is divided into 10 pieces: $\varepsilon_1, \varepsilon_2, ..., \varepsilon_{10}$. It can be observed that the budget is equally distributed in case of the Uniform allocation, whereas in the remaining budget allocation strategies, ε_i starts high and decreases as i is increased. The shape of the decrease is linear in case of Linear allocation and non-linear in case of Exponential and Step allocations.

Algorithm 2: User-side encoding and perturbation

Input : S_u, $\mathcal{D}$, $\langle \varepsilon_1, \varepsilon_2, ..., \varepsilon_\vartheta \rangle$

Output: Noisy LDP response M'_u

1 Initialize $\vartheta \times |\mathcal{D}|$ bitmatrix M'_u
2 **foreach** $i \in [1, \vartheta]$ **do**
3 **if** $i \leq |S_u|$ **then**
4 $v_u \leftarrow S_u[i]$
5 **else**
6 $v_u \leftarrow \emptyset$
7 $M'_u[i] \leftarrow \text{PERTURB}(v_u, \varepsilon_i, \mathcal{D})$ `// use AdaGRR, RAPPOR or OUE`
8 **return** M'_u

3.5 User-Side Encoding and Perturbation in Prima

The algorithm for encoding and perturbing users' data (denoted by ENCODE_PERTURB in Algorithm 1) is given in Algorithm 2. First, it initializes the bitmatrix M'_u with ϑ rows and $|\mathcal{D}|$ columns. Then, for each i between 1 and ϑ, user's sequence $S_u[i]$ is fed into an LDP protocol (AdaGRR, RAPPOR or OUE) with budget ε_i. That is, $S_u[1]$ is perturbed with budget ε_1, $S_u[2]$ is perturbed with budget ε_2, and so forth. The result of each perturbation is a bitvector, as explained in Section 3.2. On line 7, the resulting bitvector is assigned to the i'th row of M'_u. Finally, after ϑ perturbations, the user reports M'_u to the server. In Theorem 4, we prove that this process achieves sequence-level ε-LDP.

Theorem 4. *If $\varepsilon_1 + \varepsilon_2 + ... + \varepsilon_\vartheta \leq \varepsilon$, then Algorithm 2 satisfies sequence-level ε-LDP.*

Proof. Let $\mathcal{A}$ denote Algorithm 2. From the definition of sequence-level ε-LDP, we need to prove:

$$\frac{\Pr[\mathcal{A}(S_u) = M'_u]}{\Pr[\mathcal{A}(S^*_u) = M'_u]} \leq e^\varepsilon \tag{3.28}$$

For all $i \in [1, \vartheta]$, $M'_u[i]$ is constructed using an LDP protocol, which satisfies ε_i-LDP.

Formally, denoting by Ψ the LDP protocol, we know that:

$$\forall i \in [1, \vartheta] : \quad \frac{\Pr[\Psi(S_u[i]) = M'_u[i]]}{\Pr[\Psi(S^*_u[i]) = M'_u[i]]} \leq e^{\varepsilon_i} \tag{3.29}$$

Then, since Algorithm 2 considers the rows of M_u one by one and the perturbation of each row is independent from the other rows, we can write:

$$\frac{\Pr[\mathcal{A}(S_u) = M'_u]}{\Pr[\mathcal{A}(S^*_u) = M'_u]} = \prod_{i=1}^{\vartheta} \frac{\Pr[\Psi(S_u[i]) = M'_u[i]]}{\Pr[\Psi(S^*_u[i]) = M'_u[i]]} \tag{3.30}$$

$$\leq \prod_{i=1}^{\vartheta} e^{\varepsilon_i} = e^{\sum_{i=1}^{\vartheta} \varepsilon_i} = e^{\varepsilon} \tag{3.31}$$

□

3.6 Learning Markov Chain Model

The algorithm for learning the Markov chain model on the server side is given in Algorithm 3. The inputs of the algorithm are domain $\mathcal{D}$, Markov order k, noisy LDP responses (bitmatrices) received from users, and budgets $\varepsilon_1, ..., \varepsilon_\vartheta$. First, between lines 2-6, Algorithm 3 performs index-by-index summation of users' bitmatrices to compute the support (*Sup*) of each value $v_j \in \mathcal{D}$ at each index $i \in [1, \vartheta]$. After *Sup* values are computed, between lines 7-9, each $Sup[i][j]$ is given as input to the ESTIMATE function, together with the budget of the corresponding index ε_i and domain $\mathcal{D}$. Depending on which LDP protocol was used on the user side (AdaGRR, RAPPOR or OUE), the ESTIMATE function calls the appropriate estimator from Section 3.2 and the result is stored in $T[i][j]$. Here, T is a matrix for storing the estimated counts for all i, j. Then, on line 10, Algorithm 3 initializes the states of the Markov chain model $\mathcal{M}'$, as determined by $\mathcal{D}$ and k. On line 11, all possible item sequences of length k are generated and stored in a list called *seqs*. These sequences represent z of $\Pr[I_j|z]$ in Equation 3.3. Afterwards, the loop between lines 12-21 is executed to compute the transition probabilities $\Pr[I_j|z]$. Transition probabilities are all initialized as 0 (line 13). Then, the denominator of Equation 3.3 is computed (line 14) with the help of CALC_PROB function. For each transition probability to item I_j (loop lines 15-19), I_j is appended to z to obtain zI_j, and

Algorithm 3: Learn Markov Chain

Input : $\mathcal{D}, k, responses, \langle \varepsilon_1, \varepsilon_2, ..., \varepsilon_\vartheta \rangle$

Output: Markov Chain $\mathcal{M}'$

1 Initialize $\vartheta \times |\mathcal{D}|$ matrices T, Sup both full of 0s
2 **foreach** $u \in [1, \mathcal{P}]$ **do**
3 $response \leftarrow responses[u]$
4 **foreach** $i \in [1, \vartheta]$ **do**
5 **foreach** $j \in [1, |\mathcal{D}|]$ **do**
6 $Sup[i][j] \leftarrow Sup[i][j] + response[i][j]$
7 **foreach** $i \in [1, \vartheta]$ **do**
8 **foreach** $j \in [1, |\mathcal{D}|]$ **do**
9 $T[i][j] \leftarrow$ ESTIMATE$(Sup[i][j], \varepsilon_i, \mathcal{D})$ `// AdaGRR, RAPPOR or OUE`
10 Initialize the states of Markov chain model $\mathcal{M}'$
11 $seqs \leftarrow$ Generate all possible item sequences of length k `//` $|\mathcal{D}|^k$ `sequences`
12 **foreach** $z \in seqs$ **do**
13 Initialize array $probs$ with length $|\mathcal{D}|$ full of 0s
14 $denominator \leftarrow$ CALC_PROB(T, z)
15 **foreach** $j \in [1, |\mathcal{D}|]$ **do**
16 $app\_z \leftarrow$ Append I_j to z `//` I_j `is` j`'th item in` $\mathcal{D}$
17 $probs[j] \leftarrow$ CALC_PROB$(T, app_z)/denominator$
18 **if** $probs[j] < 0$ **then**
19 $probs[j] \leftarrow 0$
20 $probs \leftarrow probs/$SUM$(probs)$
21 Assign $probs$ as the transition probabilities of state z in $\mathcal{M}$'
22 **return** $\mathcal{M}'$

then the equivalent of Equation 3.3 is computed on line 17. Although transition probabilities must be non-negative by definition in a noise-free Markov chain model, due to LDP noise, the transition probability found on line 17 may be negative. Lines

Algorithm 4: CALC_PROB

Input : T, seq

Output: Probability sum p_sum

```
1 p_sum ← 0
2 foreach i ∈ [1, |T| − |seq| + 1] do
3 |   p ← 1
4 |   foreach j ∈ [1, |seq|] do
5 |   |   x ← seq[j]
6 |   |   p ← p × T[i+j][x] / Σ_{y=1}^{|D|} T[i+j][y]
7 |   p_sum ← p_sum + p
8 return p_sum
```

18-19 eliminate the possibility of negative transition probabilities. Finally, line 20 ensures that outgoing transition probabilities from any state will sum to 1, and line 21 assigns the transition probabilities of state z in the Markov chain model $\mathcal{M}'$.

The CALC_PROB function is given in Algorithm 4. When a subsequence seq occurs in a user's sequence S_u, it can occur starting at any one index or more, e.g., starting at index $S_u[1]$ and/or index $S_u[2]$ and/or $S_u[3]$, etc. CALC_PROB uses this intuition to traverse T by starting from index 1 and going up to index $|T| - |seq| + 1$ (the last possible starting index, considering the length of seq itself). For each index i, the occurrence probability of the whole sequence starting at index i is computed by iterating through each item in the sequence (lines 4-6). The probabilities are added to variable p_sum for all i (line 7). Finally, p_sum is returned (line 8).

Chapter 4

EXPERIMENTAL EVALUATION OF PRIMA

4.1 Experiment Setup and Datasets

We implemented PRIMA in Python, including all protocols, algorithms, and budget allocation strategies. In this chapter, we report the experimental evaluation of PRIMA using three datasets, four utility metrics and various ε budgets. The three datasets used in our experiments are *MSNBC*, *Rockyou* and *Taxi*.

MSNBC contains sequences of users' URL page visits from the msnbc.com website[1]. Pages are recorded at the granularity of page category (news, tech, weather, sports, etc.). There are a total of $|\mathcal{P}| = 989{,}818$ users (one sequence per user) and $|\mathcal{D}| = 17$ possible page categories.

Rockyou contains users' leaked passwords[2] as part of the Rockyou data breach. We treat each password as a sequence of characters. To limit the size of $\mathcal{D}$, we removed non-alphabetic and non-numeric characters from passwords, and ensured all passwords are lowercase. Consequently, we have $|\mathcal{D}| = 36$. There are a total of $|\mathcal{P}| = 14{,}338{,}607$ passwords in the dataset.

Taxi contains location traces of taxis driving in the city of Porto [Moreira-Matias et al., 2013]. Taxis' GPS locations were recorded in (latitude, longitude) format. Since the GPS coordinates are continuous, we discretized them as follows. First, via histogram analysis we found that more than 90% of the locations are within latitudes -8.58 to -8.68 and longitudes 41.14 to 41.18. Very few location points fall outside these ranges, which we eliminated in order to focus on the dense areas. Then, we implemented a 4x5 grid (total $|\mathcal{D}| = 20$ cells) and replaced each GPS location by

[1]UCI Machine Learning Repository: MSNBC.com Anonymous Web Data Set. `http://archive.ics.uci.edu/ml/datasets/msnbc.com+anonymous+web+data`

[2]`https://www.kaggle.com/datasets/wjburns/common-password-list-rockyoutxt`

the ID of the grid cell it fell into. Thus, each location trace was converted into a sequence of cell IDs. After this transformation, we had a total of $|\mathcal{P}| = 1{,}656{,}048$ sequences.

4.2 Utility Metrics

Let $\mathcal{M}$ denote the noise-free, gold-standard Markov chain model learned from users' sequences without any privacy guarantees. Let $\mathcal{M}'$ denote the Markov chain model learned via Prima while satisfying sequence-level ε-LDP. To preserve utility, it is desired that $\mathcal{M}'$ closely resembles $\mathcal{M}$. In order to measure this resemblance (or error), we use the following four utility metrics.

Initial Distribution Error (IDE): The initial distribution π_0 of a Markov chain is the probability distribution of that Markov chain at time 0, i.e., for each state i, the probability that the Markov chain starts out in state i. We compute the initial distributions of $\mathcal{M}$ and $\mathcal{M}'$, i.e.: $\pi_0(\mathcal{M})$ and $\pi_0(\mathcal{M}')$. Then, IDE is their average difference in l_1 norm:

$$\text{IDE} = \frac{||\pi_0(\mathcal{M}') - \pi_0(\mathcal{M})||}{\text{number of states in } \mathcal{M}} \tag{4.1}$$

Transition Probability Error (TPE): Transition probabilities constitute a key component of a Markov chain model. Let $\mathcal{M}[i][j]$ denote the transition probability from state i to j according to $\mathcal{M}$. TPE is defined as the average error across all transition probabilities in the Markov chain:

$$\text{TPE} = \frac{\sum_i \sum_j |\mathcal{M}'[i][j] - \mathcal{M}[i][j]|}{\text{number of transition probabilities}} \tag{4.2}$$

First Passage Probability Error (FPPE): Given Markov chain $\mathcal{M}$, an integer n, and two states i and j, the first passage probability is the probability of starting from i, using random walk, first passage through j to be in exactly n steps. Let $f_p(\mathcal{M}, i, j, n)$ denote a function that calculates this first passage probability. (In our experiments, we used $n = 2$.) FPPE is defined to measure the error in first passage probabilities:

$$\text{FPPE} = \frac{\sum_i \sum_j |f_p(\mathcal{M}', i, j, n) - f_p(\mathcal{M}, i, j, n)|}{(\text{number of states})^2} \tag{4.3}$$

Synthetic Dataset Error (SDE): Considering the widespread applications of Markov models to synthetic data generation in DP and LDP [Chen et al., 2012, Gursoy et al., 2018a, He et al., 2015, Wang et al., 2023a], we generate synthetic sequential datasets using our Markov chain models. When generating each sequence, we sample its first item by following the initial distribution π_0 of the Markov chain, and for the remaining items, we perform a random walk using the transition probabilities. Following this process, let D_{syn} denote the synthetic dataset generated using $\mathcal{M}$ and let D'_{syn} denote the synthetic dataset generated using $\mathcal{M}'$. We then perform frequency estimation for each item $I \in \mathcal{D}$ in D_{syn} and D'_{syn}. Let $\mathcal{F}$ and $\mathcal{F}'$ denote the results of these two frequency estimations. SDE is defined as the average difference between $\mathcal{F}$ and $\mathcal{F}'$ in l_1 norm:

$$\text{SDE} = \frac{||\mathcal{F} - \mathcal{F}'||}{|\mathcal{D}|} \tag{4.4}$$

The above metrics are closely related to potential applications of Markov chains in different contexts. For example, in the context of location traces, IDE can indicate the accuracy of modeling trip starting locations, TPE can be used to measure the error in Markov mobility modeling, and SDE can be representative of the potential utility loss in using synthetic datasets generated via Markov chains in place of real data.

4.3 Impacts of LDP Protocols

In Figure 4.1, we report experiments comparing three different LDP protocols (RAPPOR, OUE, AdaGRR) under varying datasets and ε between 0.5 and 5. In these experiments, Uniform budget allocation strategy was used. First, we observe that in general, error values decrease as ε increases. This is an intuitive outcome considering the LDP definition – as ε increases, privacy is more relaxed, therefore less noise exists in the Markov chain models. Despite the general decreasing trend in errors as ε is increased, there are a few zig-zags due to the randomized nature of LDP. We note that zig-zags are fewer and the trends are smoother on the Rockyou dataset compared to MSNBC and Taxi. The reason is because Rockyou has a much larger population size $|\mathcal{P}|$. Large $|\mathcal{P}|$ decreases the impact of randomness and increases

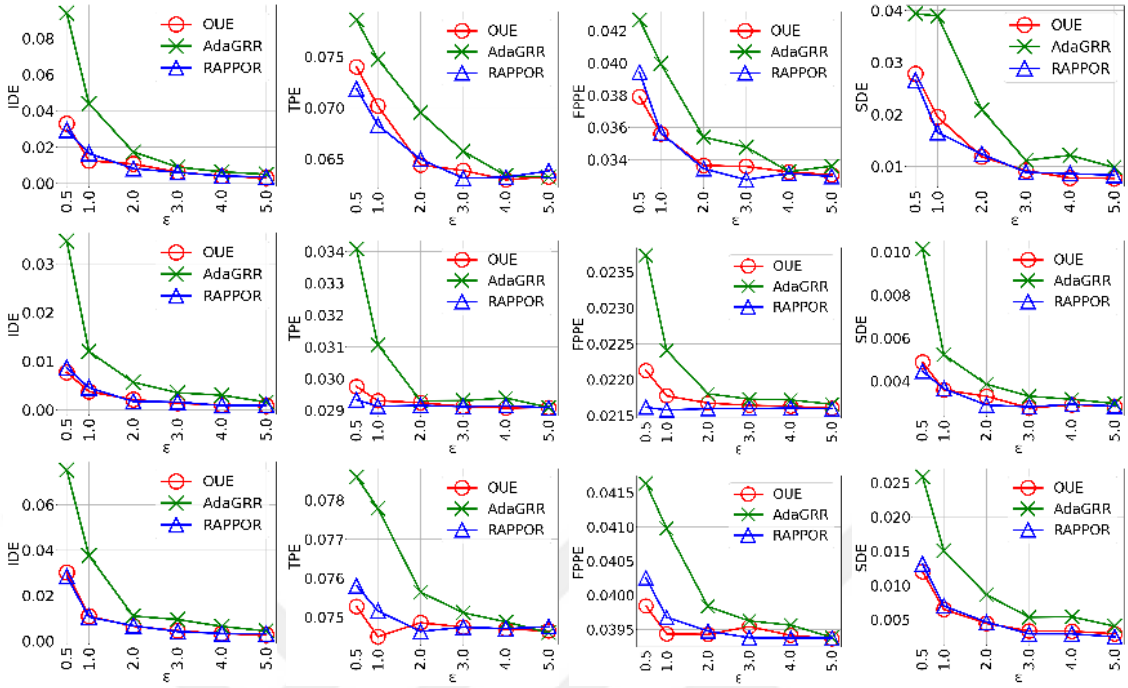

Figure 4.1: Utility impacts of different LDP protocols (RAPPOR, OUE, AdaGRR) under varying ε. Uniform budget allocation strategy was used. First row is with MSNBC dataset, second row is Rockyou dataset, third row is Taxi dataset.

Table 4.1: Execution times of Prima for different protocols and datasets ($\varepsilon = 1$, Uniform budget allocation). Times are in hh:mm:ss format.

Dataset	OUE	AdaGRR	RAPPOR
MSNBC	00:06:08	00:02:32	00:06:14
Rockyou	02:37:56	01:05:58	02:29:34
Taxi	00:13:04	00:07:44	00:13:42

estimation consistency, therefore resulting in smoother trends. Second, comparing RAPPOR, OUE and AdaGRR against each other, we observe that OUE and RAPPOR perform similarly across all utility metrics. In comparison, AdaGRR performs worse than both. Third, we observe that the difference between RAPPOR, OUE and AdaGRR is significant when ε is small (such as $\varepsilon = 0.5$), but their difference is reduced as ε becomes higher (such as $\varepsilon = 5$). At $\varepsilon = 5$, the three protocols are almost indistinguishable with respect to all utility metrics.

In Table 4.1, we report the overall execution times of PRIMA for different protocols (OUE, RAPPOR, AdaGRR). As expected, larger datasets such as Rockyou yield higher execution times. RAPPOR and OUE have similar execution times, and their time difference is small. This is because RAPPOR and OUE both work with a similar bitvector-based encoding and bit-by-bit iteration strategy, but they differ in the bit keeping and flipping probabilities. In contrast, AdaGRR is much faster than RAPPOR and OUE. On all three datasets, the execution times of AdaGRR are roughly half of RAPPOR and OUE, indicating an important speed benefit. The reason behind this speed benefit is that AdaGRR does not require bit-by-bit iteration through a bitvector, which saves $\mathcal{O}(|\mathcal{D}|)$ time cost. Combining the results in Figure 4.1 and Table 4.1, we arrive at the following trade-off: RAPPOR and OUE provide higher utility compared to AdaGRR in low ε regimes, but this utility improvement comes at the cost of higher execution time. In high ε regimes, the utility of all three protocols are similar. Yet, AdaGRR is generally faster than RAPPOR and OUE, which is expected to hold for all ε since their execution times are not impacted by ε.

4.4 Impacts of Budget Allocation Strategies

In Figure 4.2, we report experiments comparing four budget allocation strategies (Linear, Exponential, Uniform, Step) under varying ε. In these experiments, the OUE protocol was used due to its high utility. Results with only MSNBC and Rockyou datasets are reported in Figure 4.2 due to the space constraint; we remark that the results with the Taxi dataset are similar. For most utility metrics, we observe that the Step strategy performs best in low ε regimes. This is best exemplified using the IDE, TPE and SDE metrics. However, by $\varepsilon = 3$, Exponential and Linear strategies catch up, because in low ε regimes, the impact of its allocation is more pronounced since the budget is small to begin with.

Recall that the IDE metric is concerned with the initial distribution π_0 of a Markov chain model. This metric is primarily impacted by the estimation accuracy of the first items in users' sequences. Thus, the budget allocated to the first items

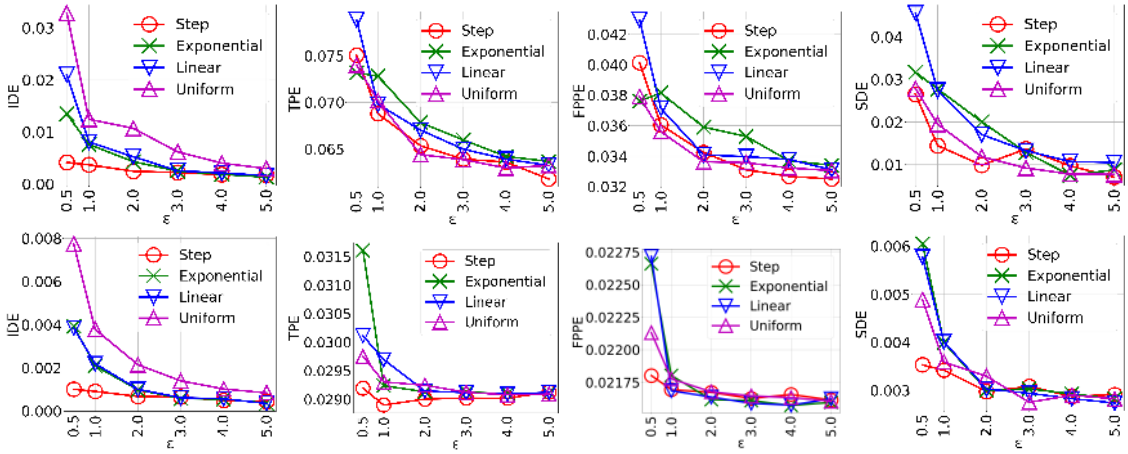

Figure 4.2: Utility impacts of different budget allocation strategies (Linear, Exponential, Uniform, Step) under varying ε. OUE protocol was used. First row is with MSNBC dataset, second row is Rockyou dataset.

(e.g., ε_1) is a key factor for IDE. Step, Linear and Exponential strategies allocate higher budgets to first few items, whereas Uniform allocates equal budget to all items. Consequently, we observe from Figure 4.2 that the Uniform strategy performs worst in terms of IDE. Similar to the above, the differences between different strategies are more pronounced in low ε regimes (e.g., $\varepsilon = 0.5$) compared to high ε regimes (e.g., when $\varepsilon = 5$, Uniform is still worse than others in terms of IDE, but their difference is less).

Finally, in many utility metrics other than IDE, Linear strategy performs worst when $\varepsilon = 0.5$. However, for $\varepsilon \geq 1$, it starts performing similarly or better than the Exponential strategy. As $\varepsilon \geq 4$, results of all budget allocation strategies start converging. In general, the Step strategy performs well across the board for most utility metrics and ε values.

Chapter 5

FEATURE SELECTION FROM MULTIDIMENSIONAL DATA UNDER LDP

In this chapter, we describe MCM and our accompanying methods for performing feature selection from multidimensional data under LDP.

5.1 Preliminaries

5.1.1 Problem Setting and Notation

We assume a setting with multidimensional data where $\mathcal{A} = \{A_1, ..., A_d\}$ denotes the set of all attributes (features). The total number of attributes is denoted by d. A_k denotes the k'th attribute, and the domain of A_k is denoted by $\Omega(A_k)$. We denote by $v_j^k \in \Omega(A_k)$ the j'th possible value in $\Omega(A_k)$, and we denote the cardinality of $\Omega(A_k)$ by $|\Omega(A_k)|$. For a set of attributes $\mathcal{S} \subseteq \mathcal{A}$, the domain of $\mathcal{S}$ is the Cartesian product of the domains of all attributes in $\mathcal{S}$, i.e.: $\Omega(\mathcal{S}) = \prod_{A_k \in \mathcal{S}} \Omega(A_k)$.

Let $\mathcal{P} = \{u_1, u_2, ..., u_{|\mathcal{P}|}\}$ denote the user population. Each user $u \in \mathcal{P}$ holds a multidimensional record r_u such that:

$$r_u = < r_u^1, r_u^2, \ldots, r_u^d > \tag{5.1}$$

where each r_u^i belongs to the domain of the corresponding feature, i.e., $r_u^i \in \Omega(A_i)$.

5.1.2 Local Differential Privacy (LDP)

Each user's record r_u is kept locally on the user's device. The data collector (server) wants to perform feature selection based on the users' records, but users are unwilling to share their records with the server due to privacy concerns. In this context, we utilize LDP to perform privacy-preserving feature selection. In LDP, each user perturbs their record on their own device and sends the perturbed version to the

server. Since perturbation occurs on the users' devices, LDP enables data collection in environments where users do not trust the server. Due to its desirable privacy properties, LDP has also been deployed in various products of tech companies such as Apple, Google, and Microsoft [Erlingsson et al., 2014, Thakurta et al., 2017, Ding et al., 2017]. We define LDP for multidimensional records as follows.

Definition 5.1.1 (ε-Local Differential Privacy)**.** Let $r_u, \tilde{r}_u \in \Omega(\mathcal{A})$ be two multidimensional records. A randomized algorithm Ψ satisfies ε-local differential privacy (ε-LDP), if and only if:

$$\forall y \in Range(\Psi): \quad \frac{\Pr[\Psi(r_u) = y]}{\Pr[\Psi(\tilde{r}_u) = y]} \leq e^{\varepsilon} \tag{5.2}$$

where $\varepsilon > 0$ is the privacy parameter and $Range(\Psi)$ denotes the set of all possible outputs of Ψ.

ε-LDP ensures that given the output y, it is not possible for the server to distinguish whether the original record was r_u or $\tilde{r}_u$ beyond the probability ratio controlled by e^{ε}. Here, ε is the privacy parameter (also known as the *privacy budget*). Lower ε yields stronger privacy.

5.1.3 Feature Selection

In case of multidimensional data, $\mathcal{A}$ may be too large for collecting or processing users' records efficiently. In addition, it may contain non-informative or redundant features which do not provide any benefit for a given learning goal or predictive task. In such scenarios, feature selection aims to select a subset of $\mathcal{A}$ that is relevant to the target variable or the task of interest, thereby discarding the features that contribute little or have no predictive value. By doing so, feature selection can significantly improve model performance, mitigate issues such as overfitting, and reduce the dimensionality of the data, thereby lowering computational costs [Blum and Langley, 1997, Li et al., 2017].

In this work, we focus on two popular feature selection methods: Information Gain (IG) and Chi-Square (χ^2). IG is derived from information theory and quantifies the reduction in entropy (uncertainty) about the target variable when a specific

feature is used to partition the data. A higher IG means that knowing the value of that feature provides a more substantial reduction in uncertainty about the target. IG is commonly used in decision trees and text classification tasks [Quinlan, 1986, Zhang et al., 2024]. χ^2 evaluates how the observed frequency of feature-target co-occurrences deviates from what would be expected if the feature and target were statistically independent. A larger χ^2 value indicates a stronger relationship between the feature and the target variable. χ^2 is especially useful for categorical features and variables [Yang et al., 1997, Li et al., 2017]. Both IG and χ^2 are well-established methods for feature selection.

Information Gain. Let $A_k \in \mathcal{A}$ and $A_l \in \mathcal{A}$ be two features with domains $\Omega(A_k)$ and $\Omega(A_l)$, respectively. The entropy of A_l is defined as:

$$H(A_l) = -\sum_{v_i^l \in \Omega(A_l)} \Pr[v_i^l] \cdot \log_2(\Pr[v_i^l]) \tag{5.3}$$

where $\Pr[v_i^l]$ is the occurrence probability of v_i^l among the user population. Then, the conditional entropy of A_l given A_k is defined as:

$$H(A_l|A_k) = -\sum_{v_j^k \Omega(A_k)} \sum_{v_i^l \in \Omega(A_l)} \Pr[v_j^k] \cdot \Pr[v_i^l|v_j^k] \cdot \log_2(\Pr[v_i^l|v_j^k]) \tag{5.4}$$

Finally, the information gain of feature A_k on feature A_l is:

$$IG(A_l, A_k) = H(A_l) - H(A_l|A_k) \tag{5.5}$$

Higher $IG(A_l, A_k)$ shows a more significant relevance between A_k and A_l.

Chi-Square (χ^2). For feature A_l with domain $\Omega(A_l)$ and for value $v_i^l \in \Omega(A_l)$, let $n_o(v_i^l)$ denote the number of users whose records contain v_i^l as the value for the A_l'th feature, i.e.:

$$n_o(v_i^l) = \text{number of users } u \in \mathcal{P} \text{ such that } r_u^l = v_i^l \tag{5.6}$$

Similarly, for feature A_k with domain $\Omega(A_k)$ and for value $v_j^k \in \Omega(A_k)$, let $n_o(v_j^k)$ denote the number of users whose records contain v_j^k as the value for the A_k'th feature, i.e.:

$$n_o(v_j^k) = \text{number of users } u \in \mathcal{P} \text{ such that } r_u^k = v_j^k \tag{5.7}$$

Furthermore, let $n_o(v_i^l, v_j^k)$ denote the number of users in $\mathcal{P}$ whose records contain v_i^l as the value for the A_l'th feature *and* v_j^k as the value for the A_k'th feature, i.e.:

$$n_o(v_i^l, v_j^k) = \text{number of users } u \in \mathcal{P} \text{ such that } r_u^l = v_i^l \text{ and } r_u^k = v_j^k \tag{5.8}$$

In other words, n_o captures the true occurrence or co-occurrence counts of v_i^l and v_j^k among the user population. On the other hand, let $n_e(v_i^l, v_j^k)$ be defined as:

$$n_e(v_i^l, v_j^k) = \frac{n_o(v_i^l) \cdot n_o(v_j^k)}{|\mathcal{P}|} \tag{5.9}$$

Here, n_e corresponds to the count that would be expected under the assumption that A_k and A_l are independent. Then, the value of the χ^2 test statistic among two features A_k and A_l can be computed as:

$$\chi^2(A_l, A_k) = \sum_{i=1}^{|\Omega(A_l)|} \sum_{j=1}^{|\Omega(A_k)|} \frac{\left(n_o(v_i^l, v_j^k) - n_e(v_i^l, v_j^k)\right)^2}{n_e(v_i^l, v_j^k)} \tag{5.10}$$

Higher value of $\chi^2(A_l, A_k)$ shows a higher dependency between the two features.

5.1.4 Problem Statement

The server would like to perform feature selection by computing IG and χ^2 between all pairs of features $A_k, A_l \in \mathcal{A}$. If users' records could be collected by the server without privacy concerns, the computation of IG and χ^2 using the aforementioned formulas would be straightforward. However, when LDP needs to be satisfied, it is not possible for the server to collect users' records in plaintext and compute IG and χ^2 in straightforward fashion, since LDP enforces each user to perturb their records locally using an LDP protocol. Hence, the server has to compute IG and χ^2 values that are estimated from noisy LDP protocol outputs. Then, the problem we study in this chapter can be stated as follows: Our goal is to propose algorithms and mechanisms such that each user's record remains protected by LDP, but the server can compute IG and χ^2 between all pairs of features $A_k, A_l \in \mathcal{A}$ with low error.

5.2 Matrix-Based Data Collection Mechanism (MCM)

A novel contribution of our work and a critical part of our solution for solving the problem stated in Section 5.1.4 is our matrix-based data collection mechanism called

MCM. Similar to existing LDP protocols from the literature, MCM consists of two components: (i) user-side encoding and perturbation to satisfy LDP locally on users' devices, and (ii) server-side estimation to recover aggregate statistics.

5.2.1 User-Side Encoding and Perturbation

The name of MCM comes from the fact that each user encodes his/her record r_u into a bitmatrix M_u. The bitmatrix M_u contains $d = |r_u|$ rows, such that for all $i \in [1, d]$, r_u^i is encoded in the i'th row $M_u[i]$. The length of the row $M_u[i]$ is equal to $|\Omega(A_i)|$. Each row contains one 1 bit depending on the value of user's r_u^i, and the remaining bits are 0. More formally, given record r_u, contents of the bitmatrix M_u are determined as:

$$\forall_{i\in[1,d]}, \forall_{j\in[1,|\Omega(A_i)|]} : \quad M_u[i][j] = \begin{cases} 1 & \text{if } r_u^i = v_j^i \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

Next, M_u is perturbed row by row to obtain a perturbed bitmatrix M'_u. Bitmatrix M'_u is identical to M_u in terms of the number of rows and columns, but its contents are different because of the perturbation. The perturbation process is similar to the application of the RAPPOR protocol to unary bitvectors, and it satisfies ε-LDP [Erlingsson et al., 2014, Wang et al., 2017, Gursoy et al., 2022]. More specifically, considering M_u contains d rows, each row is assigned the budget $\varepsilon^* = \frac{\varepsilon}{d}$, and bits in that row (say $M_u[i]$) is kept or flipped with probabilities:

$$\forall_{j\in[1,|\Omega(A_i)|]} : \quad \Pr[M'_u[i][j] = 1] = \begin{cases} p = \frac{e^{\varepsilon^*/2}}{e^{\varepsilon^*/2}+1} & \text{if } M_u[i][j] = 1 \\ q = \frac{1}{e^{\varepsilon^*/2}+1} & \text{if } M_u[i][j] = 0 \end{cases} \tag{5.12}$$

After iterating through M_u row by row in the above fashion and obtaining M'_u, the user sends M'_u to the server. We note that the two probabilities in Equation 5.12 are denoted by p and q, as they will be re-used in the derivation of server-side estimators. A step-by-step algorithmic description of the user-side encoding and perturbation process described above is given in Algorithm 5. The privacy proof of MCM's user-side encoding and perturbation is given in Theorem 5.

Theorem 5. *Algorithm 5 satisfies ε-LDP.*

Algorithm 5: User-side encoding and perturbation

Input : r_u, ε, $\mathcal{A}$, and $\Omega(A_k)$ for all $A_k \in \mathcal{A}$

Output: Perturbed bitmatrix M'_u

`/* Encoding` r_u `into bitmatrix` M_u `*/`

Initialize empty bitmatrix M_u with $|r_u|$ rows

foreach $i \in [1, |r_u|]$ **do**

  Initialize row $M_u[i]$ with length $|\Omega(A_i)|$ consisting of 0s

  Find the value $v_j^i \in \Omega(A_i)$ such that $r_u^i = v_j^i$

  Set $M_u[i][j] \leftarrow 1$

`/* Perturbation from` M_u `to` M'_u `*/`

Initialize M'_u as a deep copy of M_u

Let $\varepsilon^* \leftarrow \varepsilon / |M_u|$

foreach $i \in [1, |M_u|]$ **do**

  foreach $j \in [1, |M_u[i]|]$ **do**

$$\Pr[M'_u[i][j] = 1] = \begin{cases} p = \frac{e^{\varepsilon^*/2}}{e^{\varepsilon^*/2}+1} & \text{if } M_u[i][j] = 1 \\ q = \frac{1}{e^{\varepsilon^*/2}+1} & \text{if } M_u[i][j] = 0 \end{cases}$$

return M'_u

Proof. Let Φ denote Algorithm 5. Notice that Φ considers the rows of M_u one by one, and the perturbation of each row is independent of the other rows. Therefore, we can write:

$$\frac{\Pr[\Phi(r_u) = M'_u]}{\Pr[\Phi(\tilde{r}_u) = M'_u]} = \prod_{i=1}^{|M'_u|} \frac{\Pr[\Phi(M_u[i]) = M'_u[i]]}{\Pr[\Phi(\tilde{M}_u[i]) = M'_u[i]]} \tag{5.13}$$

Next, observe that by construction, $M_u[i]$ contains a single 1 bit and the remaining bits are 0. In other words, $M_u[i]$ is a unary bitvector. The fact that perturbing a unary bitvector according to Equation 5.12 satisfies ε^*-LDP follows from the privacy proof of RAPPOR in [Erlingsson et al., 2014]. Consequently, it holds that for each $i \in [1, |M_u|]$:

$$\frac{\Pr[\Phi(M_u[i]) = M'_u[i]]}{\Pr[\Phi(\tilde{M}_u[i]) = M'_u[i]]} \leq e^{\varepsilon^*} \tag{5.14}$$

Using this fact together with $\varepsilon^* = \frac{\varepsilon}{|M_u|} = \frac{\varepsilon}{|M'_u|}$, we can derive:

$$\prod_{i=1}^{|M'_u|} \frac{\Pr[\Phi(M_u[i]) = M'_u[i]]}{\Pr[\Phi(\tilde{M}_u[i]) = M'_u[i]]} \leq \prod_{i=1}^{|M'_u|} e^{\varepsilon^*} = e^{\varepsilon^* \cdot |M'_u|} = e^{\varepsilon} \tag{5.15}$$

□

5.2.2 Server-Side Estimation

The server receives perturbed bitmatrices M'_u from $u \in \mathcal{P}$. A key contribution and novelty of MCM is the ability to accurately estimate co-occurrence counts of values from these bitmatrices. This accurate estimation brings advantages in the computation of IG and χ^2 as well. For example, in case of χ^2, the server needs $n_o(v_i^l, v_j^k)$. From Equation 5.8, it can be observed that $n_o(v_i^l, v_j^k)$ counts the co-occurrence of v_i^l and v_j^k in the user population. Similarly, in IG, the server needs $\Pr[v_i^l | v_j^k]$ for conditional entropy calculation, which can be written as:

$$\Pr[v_i^l | v_j^k] = \frac{n_o(v_i^l, v_j^k)}{n_o(v_j^k)} \tag{5.16}$$

Hence, in both IG and $\chi 2$, more accurate estimation of the co-occurrence count $n_o(v_i^l, v_j^k)$ enables more accurate feature selection.

Say that the server wants to estimate $n_o(v_i^l, v_j^k)$. This means the server is interested in $M'_u[l][i]$ and $M'_u[k][j]$'th positions of the perturbed bitmatrices across $u \in \mathcal{P}$. To perform the necessary derivations, we introduce a few notations. We define notations for the true counts as follows:

- Let $c_{11}(v_i^l, v_j^k)$ denote the true number of users $u \in \mathcal{P}$ such that $r_u^l = v_i^l$ and $r_u^k = v_j^k$. This is equal to $n_o(v_i^l, v_j^k)$.

- Let $c_{10}(v_i^l, v_j^k)$ denote the true number of users $u \in \mathcal{P}$ such that $r_u^l = v_i^l$ but $r_u^k \neq v_j^k$.

- Let $c_{01}(v_i^l, v_j^k)$ denote the true number of users $u \in \mathcal{P}$ such that $r_u^l \neq v_i^l$ but $r_u^k = v_j^k$.

Furthermore, we define notations for the perturbed counts, i.e., results of counting from the perturbed bitmatrices M'_u, as follows:

- Let $c'_{11}(v_i^l, v_j^k)$ denote the number of users $u \in \mathcal{P}$ such that in their perturbed bitmatrix: $M'_u[l][i] = 1$ and $M'_u[k][j] = 1$.

- Let $c'_{10}(v_i^l, v_j^k)$ denote the number of users $u \in \mathcal{P}$ such that in their perturbed bitmatrix: $M'_u[l][i] = 1$ and $M'_u[k][j] = 0$.

- Let $c'_{01}(v_i^l, v_j^k)$ denote the number of users $u \in \mathcal{P}$ such that in their perturbed bitmatrix: $M'_u[l][i] = 0$ and $M'_u[k][j] = 1$.

To simplify the notation in the derivations below, we remove the (v_i^l, v_j^k) in the parantheses and simply write c_{11}, c'_{11}, c_{10}, c'_{10}, c_{01}, c'_{01}.

Using c_{11}, c_{10}, and c_{01}, we can derive the expected $\mathbb{E}[c'_{11}]$ as:

$$\mathbb{E}[c'_{11}] = c_{11}p^2 + c_{10}pq + c_{01}pq + (|\mathcal{P}| - c_{11} - c_{10} - c_{01})q^2 \tag{5.17}$$

$$= c_{11}p^2 + c_{10}pq + c_{01}pq + |\mathcal{P}|q^2 - c_{11}q^2 - c_{10}q^2 - c_{01}q^2 \tag{5.18}$$

$$= c_{11}(p^2 - q^2) + c_{10}(pq - q^2) + c_{01}(pq - q^2) + |\mathcal{P}|q^2 \tag{5.19}$$

$$= c_{11}(p^2 - q^2) + (c_{10} + c_{01})(p - q)q + |\mathcal{P}|q^2 \tag{5.20}$$

This is because across c_{11} users, the probability that both of their 1 bits are kept is p^2; across c_{10} users, the probability that their first bit is kept but the second bit is flipped is pq; across c_{01} users, the probability that their first bit is flipped but the second bit is kept is pq; and across the remaining $|\mathcal{P}| - c_{11} - c_{10} - c_{01}$ users, the probability that both of their bits are flipped is q^2.

Using a similar intuition, we can derive the expected $\mathbb{E}[c'_{10}]$ as:

$$\mathbb{E}[c'_{10}] = c_{11}pq + c_{10}p^2 + c_{01}q^2 + (|\mathcal{P}| - c_{11} - c_{10} - c_{01})pq \tag{5.21}$$

$$= c_{11}pq + c_{10}p^2 + c_{01}q^2 + |\mathcal{P}|pq - c_{11}pq - c_{10}pq - c_{01}pq \tag{5.22}$$

$$= c_{01}(q^2 - pq) + c_{10}(p^2 - pq) + |\mathcal{P}|pq \tag{5.23}$$

$$= c_{01}(q - p)q + c_{10}(p - q)p + |\mathcal{P}|pq \tag{5.24}$$

Finally, the expected $\mathbb{E}[c'_{01}]$ can be derived as:

$$\mathbb{E}[c'_{01}] = c_{11}pq + c_{10}q^2 + c_{01}p^2 + (|\mathcal{P}| - c_{11} - c_{10} - c_{01})pq \tag{5.25}$$

$$= c_{11}pq + c_{10}q^2 + c_{01}p^2 + |\mathcal{P}|pq - c_{11}pq - c_{10}pq - c_{01}pq \tag{5.26}$$

$$= c_{10}(q^2 - pq) + c_{01}(p^2 - pq) + |\mathcal{P}|pq \tag{5.27}$$

$$= c_{10}(q - p)q + c_{01}(p - q)p + |\mathcal{P}|pq \tag{5.28}$$

Next, we propose the unbiased estimators for c_{10}, c_{01}, and c_{11}. We use $\mathbb{E}[c'_{11}]$, $\mathbb{E}[c'_{10}]$, and $\mathbb{E}[c'_{01}]$ in the proofs of these estimators to show that the estimators are unbiased.

Theorem 6. *Let θ_{10} denote the estimator for c_{10}. The following estimator θ_{10} provides an unbiased estimate of c_{10}:*

$$\theta_{10} = \frac{c'_{10}p + c'_{01}q - |\mathcal{P}|pq(p+q)}{(p+q)(p-q)^2} \tag{5.29}$$

Proof. We need to show that $\mathbb{E}[\theta_{10}] = c_{10}$.

$$\mathbb{E}[\theta_{10}] = \mathbb{E}\left[\frac{c'_{10}p + c'_{01}q - |\mathcal{P}|pq(p+q)}{(p+q)(p-q)^2}\right] = \frac{\mathbb{E}[c'_{10}]p + \mathbb{E}[c'_{01}]q - |\mathcal{P}|pq(p+q)}{(p+q)(p-q)^2} \tag{5.30}$$

Earlier, we had established that:

$$\mathbb{E}[c'_{10}] = c_{01}(q - p)q + c_{10}(p - q)p + |\mathcal{P}|pq \tag{5.31}$$

$$\mathbb{E}[c'_{01}] = c_{10}(q - p)q + c_{01}(p - q)p + |\mathcal{P}|pq \tag{5.32}$$

Plugging these into Equation 5.30 and after some arithmetic operations, we get:

$$\mathbb{E}[\theta_{10}] = \frac{c_{01}(pq(q-p) + pq(q-p)) + c_{10}(p-q)(p^2-q^2) + |\mathcal{P}|pq(p-q-p+q)}{(p+q)(p-q)^2} \tag{5.33}$$

$$= \frac{c_{10}(p-q)(p-q)(p+q)}{(p+q)(p-q)^2} \tag{5.34}$$

$$= c_{10} \tag{5.35}$$

□

Theorem 7. *Let θ_{01} denote the estimator for c_{01}. The following estimator θ_{01} provides an unbiased estimate of c_{01}:*

$$\theta_{01} = \frac{c'_{01} + \theta_{10}q(p-q) - |\mathcal{P}|pq}{p(p-q)} \tag{5.36}$$

Proof. We need to show that $\mathbb{E}[\theta_{01}] = c_{01}$.

$$\mathbb{E}[\theta_{01}] = \mathbb{E}\left[\frac{c'_{01} + \theta_{10}q(p-q) - |\mathcal{P}|pq}{p(p-q)}\right] = \frac{\mathbb{E}[c'_{01}] + \mathbb{E}[\theta_{10}]q(p-q) - |\mathcal{P}|pq}{p(p-q)} \tag{5.37}$$

Earlier, we had established that:

$$\mathbb{E}[c'_{01}] = c_{10}(q-p)q + c_{01}(p-q)p + |\mathcal{P}|pq \tag{5.38}$$

$$\mathbb{E}[\theta_{10}] = c_{10} \tag{5.39}$$

Plugging these into Equation 5.37 and after some arithmetic operations, we get:

$$\mathbb{E}[\theta_{01}] = \frac{c_{10}(q-p)q + c_{01}(p-q)p + |\mathcal{P}|pq + c_{10}q(p-q) - |\mathcal{P}|pq}{p(p-q)} \tag{5.40}$$

$$= \frac{c_{10}q(q-p+p-q) + c_{01}(p-q)p}{p(p-q)} \tag{5.41}$$

$$= c_{01} \tag{5.42}$$

□

Theorem 8. *Let θ_{11} denote the estimator for c_{11}. The following estimator θ_{11} provides an unbiased estimate of c_{11}:*

$$\theta_{11} = \frac{c'_{11} - |\mathcal{P}|q^2 - (\theta_{10} + \theta_{01})(p-q)q}{p^2 - q^2} \tag{5.43}$$

Proof. We need to show that $\mathbb{E}[\theta_{11}] = c_{11}$.

$$\mathbb{E}[\theta_{11}] = \mathbb{E}\left[\frac{c'_{11} - |\mathcal{P}|q^2 - (\theta_{10} + \theta_{01})(p-q)q}{p^2 - q^2}\right] \tag{5.44}$$

$$= \frac{\mathbb{E}[c'_{11}] - |\mathcal{P}|q^2 - (\mathbb{E}[\theta_{10}] + \mathbb{E}[\theta_{01}])(p-q)q}{p^2 - q^2} \tag{5.45}$$

Earlier, we had established that:

$$\mathbb{E}[c'_{11}] = c_{11}(p^2 - q^2) + (c_{10} + c_{01})(p-q)q + |\mathcal{P}|q^2 \tag{5.46}$$

$$\mathbb{E}[\theta_{10}] = c_{10} \text{ and } \mathbb{E}[\theta_{01}] = c_{01} \tag{5.47}$$

Plugging these into Equation 5.45 and after some arithmetic operations, we get:

$$\mathbb{E}[\theta_{11}] = \frac{c_{11}(p^2 - q^2) + (c_{10} + c_{01})(p-q)q + |\mathcal{P}|q^2 - |\mathcal{P}|q^2 - (c_{10} + c_{01})(p-q)q}{p^2 - q^2} \tag{5.48}$$

$$= \frac{c_{11}(p^2 - q^2)}{p^2 - q^2} = c_{11} \tag{5.49}$$

□

Algorithm 6: Server-side estimation

Input : M'_u for all $u \in \mathcal{P}$, probabilities p and q, desired $v_i^l \in \Omega(A_l)$ and $v_j^k \in \Omega(A_k)$ for estimation

Output: Estimate of $c_{11}(v_i^l, v_j^k)$

```
Initialize c'_11, c'_10, and c'_01 as 0
foreach M'_u where u ∈ P do
    if M'_u[l][i] = 1 and M'_u[k][j] = 1 then
        c'_11 ← c'_11 + 1
    if M'_u[l][i] = 1 and M'_u[k][j] = 0 then
        c'_10 ← c'_10 + 1
    if M'_u[l][i] = 0 and M'_u[k][j] = 1 then
        c'_01 ← c'_01 + 1
Compute θ_10 using Equation 5.29
Compute θ_01 using Equation 5.36
Compute θ_11 using Equation 5.43
return θ_11
```

Making use of all of the derivations and theorems above, we construct the server-side estimation algorithm of MCM in Algorithm 6. The inputs of Algorithm 6 are the bitmatrices M'_u for all users $u \in \mathcal{P}$, the probabilities p and q that were used in perturbation, and the values $v_i^l \in \Omega(A_l)$ and $v_j^k \in \Omega(A_k)$ that we would like to estimate. The output of Algorithm 6 is the estimate of the true count $c_{11}(v_i^l, v_j^k)$, i.e., estimated number of users in the population whose records satisfy $r_u^l = v_i^l$ and $r_u^k = v_j^k$. First, Algorithm 6 goes through all users' perturbed bitmatrices and determines the values of perturbed counts c'_{11}, c'_{10}, and c'_{01}. Second, Algorithm 6 uses the estimator equations established in the above theorems to compute θ_{01}, θ_{10}, and θ_{11}. Note that the estimators have to be used in this order, since θ_{01} depends on the value of θ_{10}, and θ_{11} depends on the values of θ_{10} and θ_{01}. Finally, the estimate θ_{11} is returned.

5.3 Feature Selection Using MCM

5.3.1 End-to-End Solution Algorithm

In this section, we describe how we utilize MCM to solve the problem stated in Section 5.1.4, i.e., compute IG and χ^2 between all pairs of attributes A_k, A_l. Our end-to-end solution algorithm for solving this problem is given in Algorithm 8. The algorithm takes as input the user population $\mathcal{P}$, the privacy budget value ε, and the set of all attributes $\mathcal{A}$. It outputs $IG(A_l, A_k)$ and $\chi^2(A_l, A_k)$ for all pairs of attributes in $A_l, A_k \in \mathcal{A}$. First, each user uses MCM's user-side encoding and perturbation process (Algorithm 5) to obtain their perturbed bitmatrix M'_u. The perturbed bitmatrices are sent to the server. After the server obtains perturbed bitmatrices from all users, it uses the server-side estimation process of MCM (Algorithm 6) for each pair of attributes A_l, A_k and for each pair of values (v_i^l, v_j^k) within the attributes' domains. The resulting $c_{11}(v_i^l, v_j^k)$ values are kept in the server's memory since they will be used in the subsequent IG and χ^2 calculations.

Next, the IG calculations begin. Recall from Section 5.1.3 that the IG calculation relies on $H(A_l)$ and $H(A_l|A_k)$. To compute $H(A_l)$ and $H(A_l|A_k)$, we need the probabilities $\Pr[v_i^l]$, $\Pr[v_j^k]$ and $\Pr[v_i^l|v_j^k]$. In order to obtain these probabilities, we make use of the estimated counts c_{11}. $\Pr[v_i^l]$ is calculated as counts containing v_i^l divided by all counts; $\Pr[v_j^k]$ is calculated as counts containing v_j^k divided by all counts; and $\Pr[v_i^l|v_j^k]$ is calculated as counts containing both v_i^l and v_j^k divided by counts containing v_j^k, i.e., joint v_i^l, v_j^k divided by marginal v_j^k. After the probabilities are obtained and stored in memory, they are used to calculate $H(A_l)$, then $H(A_l|A_k)$, and finally $IG(A_l, A_k)$. Note that the same process is repeated for all different attribute pairs $A_l, A_k \in \mathcal{A}$ without the need to collect new data from the users.

Finally, we have the χ^2 calculations. Recall from Section 5.1.3 that the χ^2 calculation relies on $n_o(v_i^l, v_j^k)$ and $n_e(v_i^l, v_j^k)$. To compute $n_e(v_i^l, v_j^k)$, we also need $n_o(v_i^l)$ and $n_o(v_j^k)$. Therefore, we start by calculating $n_o(v_i^l)$ as the summation of counts containing v_i^l. We calculate $n_o(v_j^k)$ as the summation of counts containing v_j^k. We calculate the joint count $n_o(v_i^l, v_j^k)$ by directly retrieving $c_{11}(v_i^l, v_j^k)$, and we

Algorithm 7: Feature Selection Using MCM

Input : User population $\mathcal{P}$, privacy budget ε, set of attributes $\mathcal{A}$

Output: $IG(A_l, A_k)$ and $\chi^2(A_l, A_k)$ for all pairs $(A_l, A_k) \in \mathcal{A} \times \mathcal{A}$

`/* User-side encoding and perturbation */`

foreach $u \in \mathcal{P}$ **do**

  $M'_u \leftarrow$ call Algorithm 5 with user's record r_u

  User u sends M'_u to the server

`/* Server-side estimation */`

foreach *pair* $(A_l, A_k) \in \mathcal{A} \times \mathcal{A}$ **do**

  foreach (v_i^l, v_j^k) *pair such that* $v_i^l \in \Omega(A_l)$ *and* $v_j^k \in \Omega(A_k)$ **do**

    $c_{11}(v_i^l, v_j^k) \leftarrow$ call Algorithm 6 with v_i^l and v_j^k

    Store $c_{11}(v_i^l, v_j^k)$ in memory

`/* Information gain */`

foreach *pair* $(A_l, A_k) \in \mathcal{A} \times \mathcal{A}$ **do**

  foreach $v_i^l \in \Omega(A_l)$ **do**

    Compute $\Pr[v_i^l]$ as $\Pr[v_i^l] \leftarrow \frac{\sum_{v \in \Omega(A_k)} c_{11}(v_i^l, v)}{\sum_{\hat{v} \in \Omega(A_l)} \sum_{v \in \Omega(A_k)} c_{11}(\hat{v}, v)}$

  foreach $v_j^k \in \Omega(A_k)$ **do**

    Compute $\Pr[v_j^k]$ as $\Pr[v_j^k] \leftarrow \frac{\sum_{v \in \Omega(A_l)} c_{11}(v, v_j^k)}{\sum_{\hat{v} \in \Omega(A_l)} \sum_{v \in \Omega(A_k)} c_{11}(\hat{v}, v)}$

  foreach (v_i^l, v_j^k) *pair such that* $v_i^l \in \Omega(A_l)$ *and* $v_j^k \in \Omega(A_k)$ **do**

    Compute $\Pr[v_i^l | v_j^k]$ as $\Pr[v_i^l | v_j^k] \leftarrow \frac{c_{11}(v_i^l, v_j^k)}{\sum_{v \in \Omega(A_l)} c_{11}(v, v_j^k)}$

  Use Equation 5.3 to compute $H(A_l)$

  Use Equation 5.4 to compute $H(A_l|A_k)$

  Use Equation 5.5 to compute $IG(A_l, A_k)$

  Output $IG(A_l, A_k)$

calculate $n_e(v_i^l, v_j^k)$ by utilizing $n_o(v_i^l)$ and $n_o(v_j^k)$ as in Equation 5.9. Then, $n_o(v_i^l, v_j^k)$ and $n_e(v_i^l, v_j^k)$ are utilized to compute $\chi^2(A_l, A_k)$ as in Equation 5.10. Again, we note that the same process is repeated for all different attribute pairs $A_l, A_k \in \mathcal{A}$ without

Algorithm 8: Feature Selection Using MCM (continued)

... continue from Algorithm 7 ...

`/* Chi-square */`

foreach *pair* $(A_l, A_k) \in \mathcal{A} \times \mathcal{A}$ **do**

  foreach $v_i^l \in \Omega(A_l)$ **do**

    Compute $n_o(v_i^l)$ as $n_o(v_i^l) \leftarrow \sum_{v \in \Omega(A_k)} c_{11}(v_i^l, v)$

  foreach $v_j^k \in \Omega(A_k)$ **do**

    Compute $n_o(v_j^k)$ as $n_o(v_j^k) \leftarrow \sum_{v \in \Omega(A_l)} c_{11}(v, v_j^k)$

  foreach (v_i^l, v_j^k) *pair such that* $v_i^l \in \Omega(A_l)$ *and* $v_j^k \in \Omega(A_k)$ **do**

    Compute $n_o(v_i^l, v_j^k)$ as $n_o(v_i^l, v_j^k) \leftarrow c_{11}(v_i^l, v_j^k)$

    Compute $n_e(v_i^l, v_j^k)$ as $n_e(v_i^l, v_j^k) \leftarrow \frac{n_o(v_i^l) \cdot n_o(v_j^k)}{|\mathcal{P}|}$

  Use Equation 5.10 to compute $\chi^2(A_l, A_k)$

  Output $\chi^2(A_l, A_k)$

the need to collect new data from the users.

5.3.2 Time Complexity Analysis

Next, we provide a time complexity analysis of our solution in big-O notation. Recall that d denotes the number of attributes and $|\mathcal{P}|$ denotes the number of users in the population. We denote the maximum domain size by $\omega = \max_{A_k \in \mathcal{A}} \Omega(A_k)$. We perform the complexity analysis section by section, by following the sections in Algorithm 7.

User-side encoding and perturbation. Each user encodes and perturbs their own record r_u independent from the other users. Thus, we analyze the time complexity for an individual user. Note that the encoding and perturbation relies on Algorithm 5. In this algorithm, the encoding of r_u into bitmatrix M_u is $\mathcal{O}(d\omega)$. In perturbation, each bit in M_u is visited once, and there exist $\mathcal{O}(d\omega)$ bits since there are d rows and ω columns in M_u. Therefore, perturbation is also $\mathcal{O}(d\omega)$. In total, the time complexity is $\mathcal{O}(d\omega)$ per user.

Server-side estimation. There are d^2 pairs of attributes in $\mathcal{A} \times \mathcal{A}$, therefore

the outer loop iterates d^2 times. The inner loop iterates ω^2 times for each value pair. Algorithm 6 is called for each of these $d^2\omega^2$ iterations, which is $\mathcal{O}(|\mathcal{P}|)$ since it linearly iterates over users' perturbed bitmatrices. Thus, the total complexity of this section is $\mathcal{O}(d^2\omega^2|\mathcal{P}|)$.

Information gain. There are d^2 pairs of attributes in $\mathcal{A} \times \mathcal{A}$, therefore the outer loop iterates d^2 times. The first inner loop iterates ω times. Within each iteration, computing the numerator in the fraction is $\mathcal{O}(\omega)$ and the denominator is $\mathcal{O}(\omega^2)$. Hence, as a whole, the first inner loop is $\mathcal{O}(\omega^3)$. The second inner loop is also $\mathcal{O}(\omega^3)$; the calculation is similar to the previous loop. The third inner loop iterates ω^2 times and computing the fraction within the loop is $\mathcal{O}(\omega)$, therefore the third inner loop is also $\mathcal{O}(\omega^3)$. Then, computing $H(A_l)$ is $\mathcal{O}(\omega)$, computing $H(A_l|A_k)$ is $\mathcal{O}(\omega^2)$, and computing $IG(A_l, A_k)$ is $\mathcal{O}(1)$. In total, the time complexity of this section becomes $\mathcal{O}(d^2\omega^3)$.

Chi-square. There are d^2 pairs of attributes in $\mathcal{A} \times \mathcal{A}$, therefore the outer loop iterates d^2 times. The first inner loop iterates ω times and each iteration is $\mathcal{O}(\omega)$, therefore the loop as a whole is $\mathcal{O}(\omega^2)$. The second inner loop is also $\mathcal{O}(\omega^2)$; the calculation is similar to the previous loop. The third inner loop iterates ω^2 times and each iteration is $\mathcal{O}(1)$; thus, the third inner loop is also $\mathcal{O}(\omega^2)$. Using Equation 5.10 to compute $\chi^2(A_l, A_k)$ is $\mathcal{O}(\omega^2)$. In total, the time complexity of this section becomes $\mathcal{O}(d^2\omega^2)$.

Analysis and discussion. Overall, the time complexity per user is $\mathcal{O}(d\omega)$, which is linear in both d and ω. This complexity shows that the user-side is efficient and scalable. On the server side, estimation is $\mathcal{O}(d^2\omega^2|\mathcal{P}|)$, information gain computation is $\mathcal{O}(d^2\omega^3)$, and chi-square computation is $\mathcal{O}(d^2\omega^2)$. In practice, we note that $|\mathcal{P}|$ is much larger than ω, e.g., $|\mathcal{P}|$ is in the order of ten thousands or hundred thousands whereas ω is in the order of tens or hundreds. Thus, we can conclude that the server-side time complexity is $\mathcal{O}(d^2\omega^2|\mathcal{P}|)$. From this, we observe that the complexity is quadratic in terms of the number of attributes and attribute domain sizes, whereas it is linear in the number of users.

5.3.3 Space Complexity Analysis

Next, we provide a space (memory) complexity analysis of our solution in big-O notation. Similar to the time complexity analysis, we perform the space complexity analysis section by section.

User-side encoding and perturbation. The size of the bitmatrices M_u and M'_u is $\mathcal{O}(d\omega)$. Therefore, the space complexity is $\mathcal{O}(d\omega)$ per user.

Server-side estimation. The server receives perturbed bitmatrices, each of size $\mathcal{O}(d\omega)$, from $|\mathcal{P}|$ users. Thus, there is $\mathcal{O}(d\omega|\mathcal{P}|)$ space complexity. In addition, the outer loop iterates d^2 times and the inner loop iterates ω^2 times. For a total of $d^2\omega^2$ iterations, a different c_{11} count is stored in memory. Thus, the total space complexity of this section becomes $\mathcal{O}(d\omega|\mathcal{P}| + d^2\omega^2)$.

Information gain. There are three inner loops within each iteration of the outer loop. The first inner loop has a space complexity of $\mathcal{O}(\omega)$ since each $\Pr[v_i^l]$ needs to be stored for subsequent computations. The second inner loop also has a space complexity of $\mathcal{O}(\omega)$. The third inner loop has a complexity of $\mathcal{O}(\omega^2)$ because $\Pr[v_i^l|v_j^k]$ needs to be stored for each (v_i^l, v_j^k) pair. Rest of the computations are $\mathcal{O}(1)$. Thus, assuming the output $IG(A_l, A_k)$ does not have to be stored in memory, the overall complexity of this section is $\mathcal{O}(\omega^2)$.

Chi-square. Again, there are three inner loops within each iteration of the outer loop. The first and second inner loops each have a space complexity of $\mathcal{O}(\omega)$ since each $n_o(v_i^l)$ and $n_o(v_j^k)$ needs to be stored for subsequent computations. The third inner loop has a complexity of $\mathcal{O}(\omega^2)$ since each $n_o(v_i^l, v_j^k)$ and $n_e(v_i^l, v_j^k)$ need to be stored for subsequent computations, and there are ω^2 instances of each. Thus, assuming the output $\chi^2(A_l, A_k)$ does not have to be stored in memory, the overall complexity of this section is $\mathcal{O}(\omega^2)$.

Analysis and discussion. Overall, the space complexity per user is $\mathcal{O}(d\omega)$, which is the same as the time complexity, and it is linear in both d and ω. This is beneficial for efficiency and scalability. On the server side, we observe that the main space complexity (and therefore the storage cost) arises from the server-side estimation phase, which is $\mathcal{O}(d\omega|\mathcal{P}| + d^2\omega^2)$. Computing information gain and chi-

square have much lower complexities of $\mathcal{O}(\omega^2)$ each. In our experiments, we observed that a commodity laptop with 16 GB memory can comfortably serve as the server machine, and it is more than enough to handle the server-side storage cost.

Chapter 6

EXPERIMENTAL EVALUATION OF MCM

6.1 Experiment Setup

We implemented MCM and our feature selection algorithm based on MCM in Python. In this chapter, we report their experimental evaluation using three datasets, two utility metrics, and various ε budgets. Furthermore, our solution is compared against the state of the art solution for feature selection under LDP, called LDP-FS [Alishahi et al., 2022].

6.1.1 Datasets

The three datasets used in our experiments are *Adult*, *Nursery*, and *Mushroom*. Among these datasets, Nursery and Mushroom were also used in the evaluation of LDP-FS.

Adult is a widely used dataset in the data privacy literature. It can be downloaded from the UCI ML Repository[1] and contains demographic and employment-related information about individuals. From the attributes that originally exist in Adult, we removed fnlwgt, education-num, capital-gain, and capital-loss. We also removed records that had missing values for one or more attributes. In the end, we had 45.222 remaining records.

Nursery is a dataset that was originally developed to rank applications for nursery schools. It contains 12.958 records and 7 categorical attributes. It can also be downloaded from the UCI ML Repository[2].

Mushroom contains 8.124 records corresponding to samples of mushrooms from

[1] `https://archive.ics.uci.edu/dataset/2/adult`

[2] `https://archive.ics.uci.edu/dataset/76/nursery`

the Agaricus and Lepiota families[3]. There are a total of 21 attributes which correspond to the characteristics of mushrooms such as cap shape, cap surface, color, and odor. The dataset is used for building classification models to predict whether a mushroom is poisonous or not.

6.1.2 Utility Metrics

For a pair of attributes $A_l, A_k \in \mathcal{A}$, let $IG(A_l, A_k)$ and $\chi^2(A_l, A_k)$ denote their ground truth (real) information gain and chi-square values respectively. Furthermore, let $\widehat{IG}(A_l, A_k)$ and $\widehat{\chi}^2(A_l, A_k)$ denote the noisy information gain and chi-square values that are obtained after LDP. To measure the error between $IG(A_l, A_k)$ and $\widehat{IG}(A_l, A_k)$, as well as between $\chi^2(A_l, A_k)$ and $\widehat{\chi}^2(A_l, A_k)$, we utilize the Root Mean Squared Error (RMSE) metric. RMSE for information gain is defined as:

$$\text{RMSE}_{IG} = \sqrt{\sum_{A_l \in \mathcal{A}} \sum_{A_k \in \mathcal{A}} \frac{(IG(A_l, A_k) - \widehat{IG}(A_l, A_k))^2}{|\mathcal{A}|^2}} \tag{6.1}$$

Similarly, RMSE for χ^2 is defined as:

$$\text{RMSE}_{\chi^2} = \sqrt{\sum_{A_l \in \mathcal{A}} \sum_{A_k \in \mathcal{A}} \frac{(\chi^2(A_l, A_k) - \widehat{\chi}^2(A_l, A_k))^2}{|\mathcal{A}|^2}} \tag{6.2}$$

6.1.3 Comparison with LDP-FS

We compare our proposed solution with LDP-FS, the state-of-the-art solution for feature selection under LDP [Alishahi et al., 2022]. Note that our solution is able to compute information gains and χ^2 values for all pairs of attributes $A_l, A_k \in \mathcal{A}$ in a single data collection with MCM, utilizing the full privacy budget of ε. In contrast, LDP-FS identifies one attribute as the class attribute (denoted by $\mathcal{C}$), and computes $IG(A_k, \mathcal{C})$ and $\chi^2(A_k, \mathcal{C})$ for attributes $A_k \in \mathcal{A}$. Thus, unlike our solution, LDP-FS does not compute information gains and χ^2 values among all pairs of attributes by default. To ensure a fair comparison between our solution and LDP-FS, we rerun LDP-FS d times, each time with a different class attribute $\mathcal{C}$ selected from $\mathcal{A}$.

[3] `https://archive.ics.uci.edu/dataset/73/mushroom`

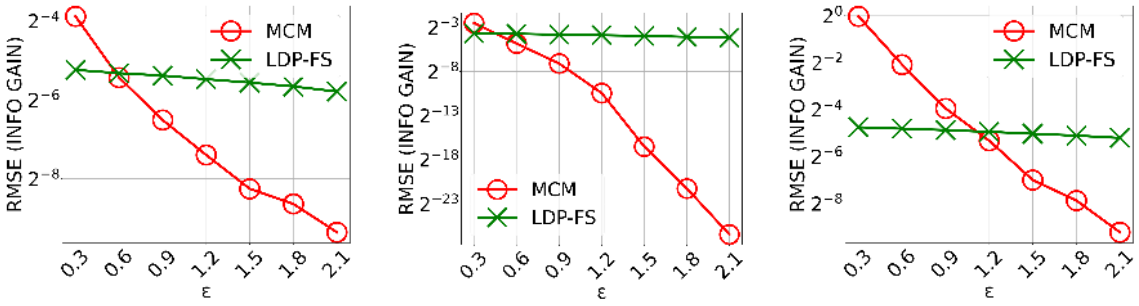

Figure 6.1: RMSE of IG with our solution (MCM) versus LDP-FS under varying ε between 0.3 and 2.1. First plot is with the Nursery dataset, second plot is with the Mushroom dataset, third plot is with the Adult dataset.

This way, we are able to obtain information gains and χ^2 values among all pairs of attributes using LDP-FS. To ensure that ε-LDP is preserved across all d executions of LDP-FS, each execution is performed with a privacy budget of ε/d.

We also note that LDP-FS can use multiple LDP protocols internally: OLH, OUE, THE, HR, CMS, or RAPPOR. The experiments in [Alishahi et al., 2022] indicate that RAPPOR performs better than other protocols, i.e., it usually yields the lowest errors. Therefore, in our experiments, when comparing our solution with LDP-FS, we use LDP-FS with RAPPOR so that we compare with the best performance of LDP-FS.

6.2 Main Results and Discussion

In Figure 6.1, we provide the results of comparing our solution (denoted by MCM in the plots) with LDP-FS in terms of information gain. First, we observe that in both MCM and LDP-FS, RMSE values decrease as ε increases. This is an intuitive outcome considering the LDP definition; as ε increases, privacy is more relaxed, therefore less noise is added. Second, we observe that for lower ε values such as 0.3 or 0.6, there are instances where our solution performs worse than LDP-FS. However, as ε is increased, our solution quickly catches up with LDP-FS, and it achieves significantly lower errors than LDP-FS for higher ε such as $\varepsilon \geq 1.5$. Third, we obseve that as ε increases, errors of LDP-FS decrease slowly, whereas errors of our solution decrease significantly. Overall, for a large majority of the ε values, our

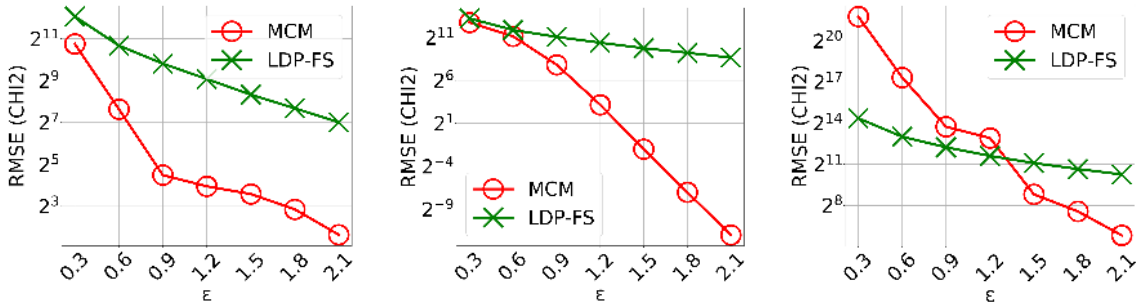

Figure 6.2: RMSE of χ^2 with our solution (MCM) versus LDP-FS under varying ε between 0.3 and 2.1. First plot is with the Nursery dataset, second plot is with the Mushroom dataset, third plot is with the Adult dataset.

solution yields lower errors than LDP-FS. In few instances, especially under strict privacy regimes (i.e., low ε), LDP-FS can be better. We highlight that when $\varepsilon > 1.5$, our solution is substantially better than LDP-FS (notice that the y-axis of the plots in Figure 6.1 are in logarithmic scale).

In Figure 6.2, we provide the results of comparing our solution (again denoted by MCM in the plots) with LDP-FS in terms of χ^2. Compared to Figure 6.1, we see similar trends in Figure 6.2. Our solution outperforms LDP-FS for the large majority of ε values. Furthermore, the difference between our solution and LDP-FS is even more pronounced in the χ^2 metric compared to IG. On the other hand, we note that LDP-FS outperforms our solution on the Adult dataset under low ε. We believe that this is caused by the fact that Adult is significantly larger than the other datasets, enabling LDP-FS to achieve high estimation accuracy despite ε/d budget division. This intuition is also supported by the fact that Mushroom is the dataset with the largest number of attributes (high d), and it is the dataset where our solution outperforms LDP-FS most clearly in terms of both IG and χ^2. Another interesting question that arises from Figures 6.1 and 6.2 is: Why does our solution yield comparable errors to LDP-FS when ε is low, but much better errors than LDP-FS when ε is high? We believe that this is because when ε is low, both LDP-FS and MCM are extremely noisy. Hence, MCM's server-side estimation results c_{11} are quite inaccurate. This inaccuracy propagates through all of the IG and χ^2 calculations, and yield high RMSEs.

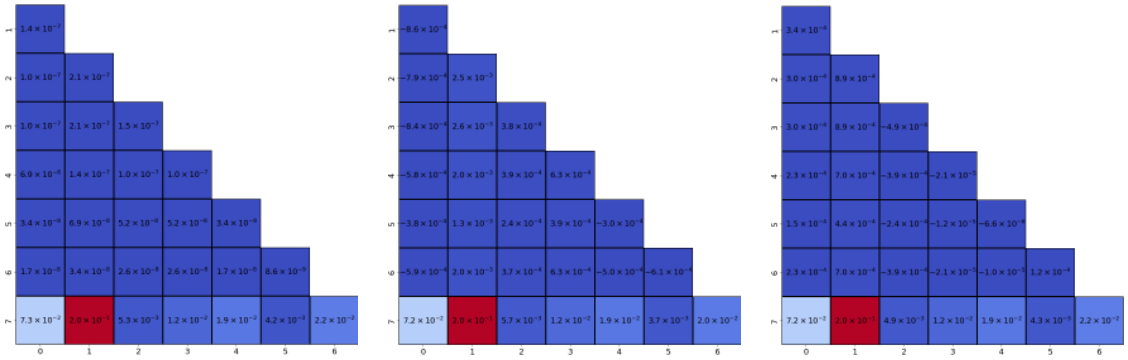

Figure 6.3: $|\mathcal{A}| \times |\mathcal{A}|$ heatmaps of IG values between attributes in the Nursery dataset. Original heatmap (ground truth) on the left, $\varepsilon = 1.5$ in the middle, $\varepsilon = 2.1$ on the right.

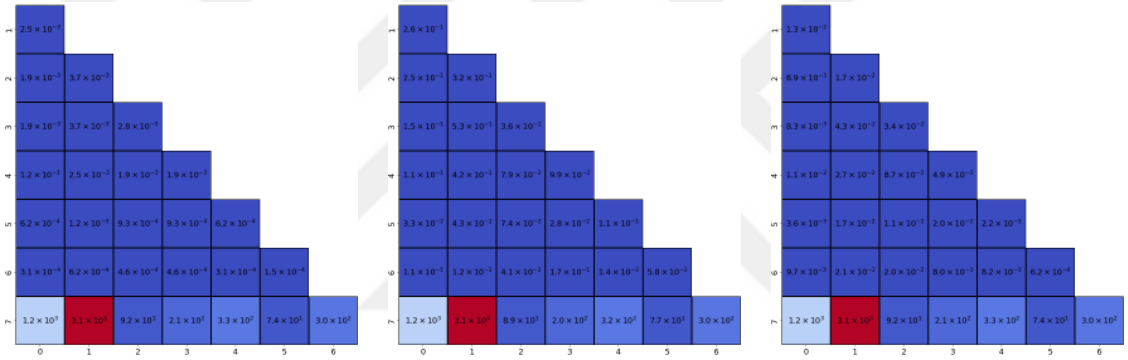

Figure 6.4: $|\mathcal{A}| \times |\mathcal{A}|$ heatmaps of χ^2 values between attributes in the Nursery dataset. Original heatmap (ground truth) on the left, $\varepsilon = 1.5$ in the middle, $\varepsilon = 2.1$ on the right.

6.3 Heatmap Analysis for IG and χ^2

Next, we perform a heatmap analysis of IG and χ^2 values between all attributes in the datasets, to observe if the relative IG and χ^2 between attributes are preserved. First, we draw an $|\mathcal{A}|\times|\mathcal{A}|$ heatmap of IG values (or χ^2 values) between all attributes from the raw data, which serves as the ground truth. Then, we draw the same heatmaps using noisy IG (or χ^2 values) obtained from our solution under LDP. This is repeated for two different ε budgets: $\varepsilon = 1.5$ and 2.1. IG and χ^2 heatmaps for the Nursery dataset are given in Figures 6.3 and 6.4; IG and χ^2 heatmaps for the Adult dataset are given in Figures 6.5 and 6.6. In all heatmaps, darker colored cells indicate lower IG and χ^2 between the corresponding attributes, whereas brighter

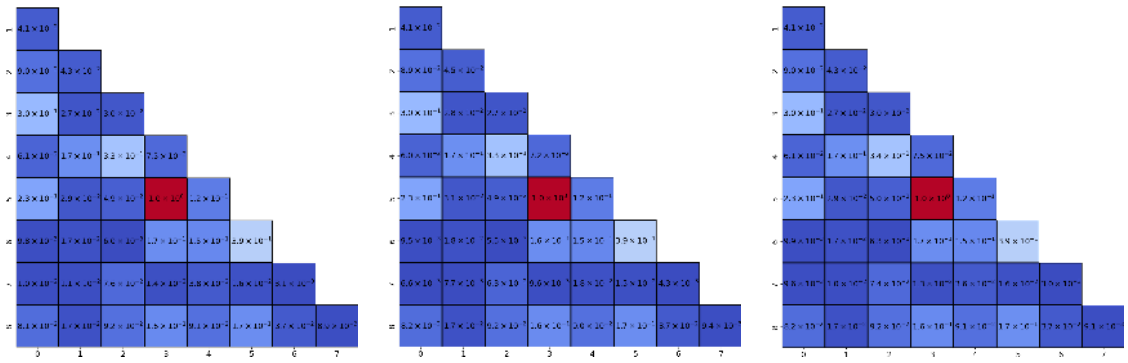

Figure 6.5: $|\mathcal{A}| \times |\mathcal{A}|$ heatmaps of IG values between attributes in the Adult dataset. Original heatmap (ground truth) on the left, $\varepsilon = 1.5$ in the middle, $\varepsilon = 2.1$ on the right.

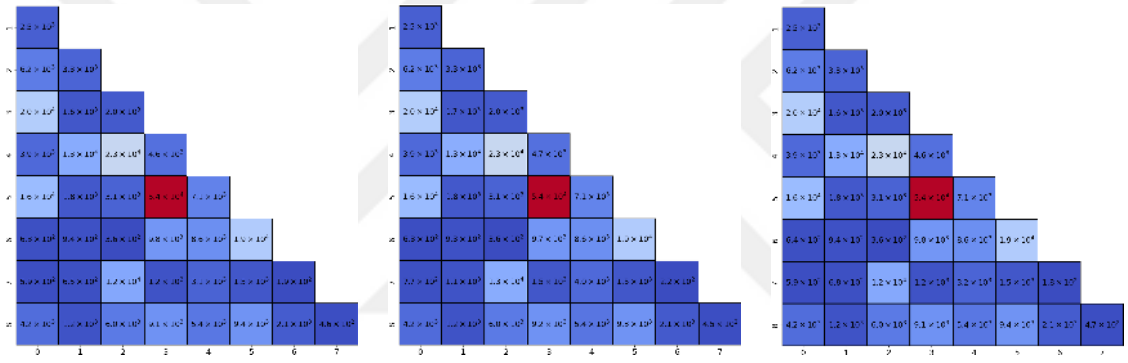

Figure 6.6: $|\mathcal{A}| \times |\mathcal{A}|$ heatmaps of χ^2 values between attributes in the Adult dataset. Original heatmap (ground truth) on the left, $\varepsilon = 1.5$ in the middle, $\varepsilon = 2.1$ on the right.

colored cells indicate higher IG and χ^2. The IG and χ^2 values themselves are written inside the cells.

We make several observations from the figures. First, studying the ground truth heatmaps only, we observe that the ground truth IG heatmap is similar to the χ^2 heatmap for the Nursery dataset. Similarly, the ground truth IG heatmap is similar to the χ^2 heatmap for the Adult dataset. This shows that the correlations and predictive values of attributes discovered by IG and χ^2 agree with one another. Second, we observe that the errors in the noisy heatmaps (with $\varepsilon = 1.5$ and 2.1) decrease as ε is increased. For example, in Figure 6.3, the ground truth of $IG(A_6, A_7)$ = 0.022 is estimated correctly when $\varepsilon = 2.1$, but there is an error of 0.002 when ε

Table 6.1: Execution times of our solution versus LDP-FS for all datasets. Times are given in seconds.

	User-Side	Server-Side	Total (Our Solution)	LDP-FS
Nursery	0.17	3.79	3.96	27.76
Mushroom	0.35	43.24	43.59	57.15
Adult	1.09	201.17	202.26	102.54

= 1.5. This is caused by LDP noise.

Third, an important observation is that the noisy heatmaps are similar to the original heatmaps in terms of colors. The actual values written inside the cells may change because of LDP noise, but the trends regarding which two attributes are more correlated versus which attributes are uncorrelated are preserved despite LDP. This is a positive finding which shows the strength of our solution. For example, if the data collector were to rank the attributes in terms of χ^2 or IG and eliminate low-ranked ones, the data collector's eliminations would be correct compared to the ground truth.

6.4 Execution Time Analysis

In this section, we analyze the execution times of our solution and compare them with the execution times of LDP-FS. For the execution time experiments, we used a fixed $\varepsilon = 1.2$. We note that changing the ε value does not have a noticeable impact on the execution times. The results are shown in Table 6.1. For our solution, we measure the execution times of the following components separately: (i) user-side encoding and perturbation, (ii) server-side estimation and computation of IG and χ^2, and (iii) total execution time, i.e., all of Algorithm 7. For LDP-FS, we measure the total execution time.

First, we observe that the user-side execution times of our solution are short, e.g., 0.17 seconds for Nursery, 0.35 seconds for Mushroom, and 1.09 seconds for Adult. On the other hand, the server-side execution times are much longer: 3.79, 43.24, and 201.17 seconds for the three datasets, respectively. From the time complexity

analysis, we know that the user-side complexity is $\mathcal{O}(d\omega)$ whereas the server-side complexity is $\mathcal{O}(d^2\omega^2|\mathcal{P}|)$. Given these complexities, it is normal that the server-side execution times are much higher than user-side execution times.

Nevertheless, the total execution times of our solution are shorter than LDP-FS on two of the datasets: Nursery and Mushroom. We observe that our solution is faster than LDP-FS by a large margin on the Nursery dataset. The difference between our solution and LDP-FS is smaller on the Mushroom dataset, but our solution is still faster. However, LDP-FS is faster than our solution on the Adult dataset. We believe that this behavior is due to the domain sizes of the attributes in the datasets. The mean domain size of attributes in Nursery is 3.5, whereas it is 5.17 in Mushroom and 11.2 in Adult. From these, we can conclude that as the domain size increases, our solution's execution time suffers more than LDP-FS.

Chapter 7

CONCLUSION

The growing importance of privacy-preserving data collection and analysis has driven extensive research into Local Differential Privacy (LDP) as a standard for ensuring privacy protection. In this thesis, we contributed to advancing the state-of-the-art in the applications of LDP to two fundamental data types: sequential data and multidimensional data. We proposed two novel mechanisms, PRIMA and MCM, to enable privacy-preserving Markov chain learning and feature selection, respectively. Through theoretical analysis and extensive experimentation, we demonstrated the utility and efficiency of our proposed approaches.

We proposed PRIMA to enable the learning of Markov chain models under sequence-level LDP. Key contributions of PRIMA include: (i) the development of AdaGRR, a novel extension of the GRR protocol, which enables efficient bitvector-based perturbation, (ii) a server-side estimation procedure and specialized algorithms for recovering Markov transition probabilities from perturbed sequential data, (iii) budget allocation strategies to improve utility, and (iv) a comprehensive experimental evaluation, enabling recommendations regarding the usage of PRIMA's parameters and budget allocation strategies. Our experiments showed that OUE and RAPPOR protocols generally provided better utility, while AdaGRR, though slightly less accurate at small ε, offered significant speed benefits. We also found that different budget allocation strategies impacted utility differently, with the Step strategy performing well across most settings.

We proposed MCM and accompanying algorithms to address the challenge of feature selection from multidimensional data under LDP. Key contributions of MCM and its accompanying algorithms include: (i) a novel bitmatrix-based LDP mechanism that enables server-side estimations of co-occurrence counts between all value pairs, (ii) server-side algorithms to compute two popular feature selection metrics,

IG and χ^2, between all pairs of attributes, and (iii) an experimental evaluation which shows that our algorithms outperform a state-of-the-art LDP feature selection solution, LDP-FS [Alishahi et al., 2022]. Unlike LDP-FS which focuses on feature-class relationships, our solution is capable of capturing pairwise IG and χ^2 between all pairs of attributes using a single data collection from users.

The methods proposed in this thesis contribute to making LDP applicable to a broader range of real-world data types and tasks, expanding its usability in privacy-preserving data analytics. Our work has implications for sectors such as healthcare, finance, cybersecurity, and web analytics, where sequential and multidimensional data are frequently found and protecting user data is crucial. Yet, there remain several future work directions. First, we plan to improve the scalability and efficiency of PRIMA for higher-order Markov chains. Second, our results showed that while AdaGRR is faster than RAPPOR and OUE, its utility can be lower under small ε. We plan to derive the variance of AdaGRR to obtain insights regarding AdaGRR's expected ℓ_2 error, and improve the utility of AdaGRR in general. Third, we will evaluate different budget allocation strategies in PRIMA with protocols other than OUE, to verify if our budget allocation-related results are generalizable to multiple protocols. Fourth, we plan to apply MCM to feature selection methods other than IG and χ^2.

BIBLIOGRAPHY

[app, 2020] (2020). Apple – learning with privacy at scale. `https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf`.

[Alishahi et al., 2022] Alishahi, M., Moghtadaiee, V., and Navidan, H. (2022). Add noise to remove noise: Local differential privacy for feature selection. *Computers & Security*, 123:102934.

[Arcolezi et al., 2021] Arcolezi, H. H., Couchot, J.-F., Al Bouna, B., and Xiao, X. (2021). Random sampling plus fake data: Multidimensional frequency estimates with local differential privacy. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 47–57.

[Arcolezi et al., 2024] Arcolezi, H. H., Couchot, J.-F., Al Bouna, B., and Xiao, X. (2024). Improving the utility of locally differentially private protocols for longitudinal and multidimensional frequency estimates. *Digital Communications and Networks*, 10(2):369–379.

[Blum and Langley, 1997] Blum, A. L. and Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271.

[Chen et al., 2023] Chen, B., Leahy, K., Jones, A., and Hale, M. (2023). Differential privacy for symbolic systems with application to markov chains. *Automatica*, 152:110908.

[Chen et al., 2012] Chen, R., Acs, G., and Castelluccia, C. (2012). Differentially private sequential data publication via variable-length n-grams. Proceedings of

the 2012 ACM SIGSAC Conference on Computer and Communications Security, page 638–649. ACM.

[Cheng et al., 2016] Cheng, Y., Qiao, Y., and Yang, J. (2016). An improved markov method for prediction of user mobility. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 394–399. IEEE.

[Cormode et al., 2018a] Cormode, G., Jha, S., Kulkarni, T., Li, N., Srivastava, D., and Wang, T. (2018a). Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1655–1658. ACM.

[Cormode et al., 2018b] Cormode, G., Kulkarni, T., and Srivastava, D. (2018b). Marginal release under local differential privacy. In *Proceedings of the 2018 International Conference on Management of Data*, pages 131–146. ACM.

[Cunningham et al., 2021] Cunningham, T., Cormode, G., Ferhatosmanoglu, H., and Srivastava, D. (2021). Real-world trajectory sharing with local differential privacy. *Proceedings of the VLDB Endowment*, 14(11):2283–2295.

[da Costa Filho and Machado, 2023] da Costa Filho, J. S. and Machado, J. C. (2023). Felip: A local differentially private approach to frequency estimation on multidimensional datasets. In *Proceedings of the 26th International Conference on Extending Database Technology (EDBT)*, pages 671–683.

[Ding et al., 2017] Ding, B., Kulkarni, J., and Yekhanin, S. (2017). Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580.

[Du et al., 2021a] Du, L., Zhang, Z., Bai, S., Liu, C., Ji, S., Cheng, P., and Chen, J. (2021a). Ahead: adaptive hierarchical decomposition for range query under local differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1266–1288.

[Du et al., 2021b] Du, R., Ye, Q., Fu, Y., and Hu, H. (2021b). Collecting high-dimensional and correlation-constrained data with local differential privacy. In *18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE.

[Erlingsson et al., 2014] Erlingsson, Ú., Pihur, V., and Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1054–1067. ACM.

[Errounda and Liu, 2021] Errounda, F. Z. and Liu, Y. (2021). Collective location statistics release with local differential privacy. *Future Generation Computer Systems*, 124:174–186.

[Fan et al., 2014] Fan, L., Bonomi, L., Xiong, L., and Sunderam, V. (2014). Monitoring web browsing behavior with differential privacy. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, page 177–188. Association for Computing Machinery.

[Fine et al., 1998] Fine, S., Singer, Y., and Tishby, N. (1998). The hierarchical hidden markov model: analysis and applications. *Machine Learning*, 32(1):41–62.

[Gagniuc, 2017] Gagniuc, P. (2017). *Markov Chains: From Theory to Implementation and Experimentation*.

[Gales et al., 2008] Gales, M., Young, S., et al. (2008). The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304.

[Gambs et al., 2012] Gambs, S., Killijian, M.-O., and del Prado Cortez, M. N. (2012). Next place prediction using mobility markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, pages 1–6.

[Gursoy et al., 2022] Gursoy, M. E., Liu, L., Chow, K.-H., Truex, S., and Wei, W. (2022). An adversarial approach to protocol analysis and selection in local differential privacy. *IEEE Transactions on Information Forensics and Security*, 17:1785–1799.

[Gursoy et al., 2018a] Gursoy, M. E., Liu, L., Truex, S., and Yu, L. (2018a). Differentially private and utility preserving publication of trajectory data. *IEEE Transactions on Mobile Computing*, 18(10):2315–2329.

[Gursoy et al., 2018b] Gursoy, M. E., Liu, L., Truex, S., Yu, L., and Wei, W. (2018b). Utility-aware synthesis of differentially private and attack-resilient location traces. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 196–211.

[Gursoy et al., 2020] Gursoy, M. E., Rajasekar, V., and Liu, L. (2020). Utility-optimized synthesis of differentially private location traces. In *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 30–39. IEEE.

[Gursoy et al., 2019] Gursoy, M. E., Tamersoy, A., Truex, S., Wei, W., and Liu, L. (2019). Secure and utility-aware data collection with condensed local differential privacy. *IEEE Transactions on Dependable and Secure Computing*.

[He et al., 2015] He, X., Cormode, G., Machanavajjhala, A., Procopiuc, C., and Srivastava, D. (2015). Dpt: differentially private trajectory synthesis using hierarchical reference systems. *Proceedings of the VLDB Endowment*, 8(11):1154–1165.

[Li et al., 2017] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2017). Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45.

[Liu et al., 2023] Liu, G., Tang, P., Hu, C., Jin, C., Guo, S., Stoyanovich, J., Teubner, J., Mamoulis, N., Pitoura, E., and Mühlig, J. (2023). Multi-dimensional data

publishing with local differential privacy. In *Proceedings of the 26th International Conference on Extending Database Technology (EDBT)*, pages 183–194.

[Makhlouf et al., 2024] Makhlouf, K., Arcolezi, H. H., Zhioua, S., Brahim, G. B., and Palamidessi, C. (2024). On the impact of multi-dimensional local differential privacy on fairness. *Data Mining and Knowledge Discovery*, pages 1–24.

[Meyn and Tweedie, 2012] Meyn, S. P. and Tweedie, R. L. (2012). *Markov chains and stochastic stability*. Springer Science & Business Media.

[Mor et al., 2020] Mor, B., Garhwal, S., and Loura, A. (2020). A systematic review of hidden markov models and their applications. *Archives of Computational Methods in Engineering*, 28.

[Moreira-Matias et al., 2013] Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., and Damas, L. (2013). Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402.

[Murakami and Kawamoto, 2019] Murakami, T. and Kawamoto, Y. (2019). Utility-Optimized local differential privacy mechanisms for distribution estimation. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1877–1894, Santa Clara, CA. USENIX Association.

[Qian et al., 2023] Qian, Q., Ye, Q., Hu, H., Huang, K., Chan, T. T.-L., and Li, J. (2023). Collaborative sampling for partial multi-dimensional value collection under local differential privacy. *IEEE Transactions on Information Forensics and Security*, 18:3948–3961.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

[Ren et al., 2018] Ren, X., Yu, C.-M., Yu, W., Yang, S., Yang, X., McCann, J. A., and Philip, S. Y. (2018). Lopub: High-dimensional crowdsourced data publication

with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 13(9):2151–2166.

[Shen and Yu, 2013] Shen, E. and Yu, T. (2013). Mining frequent graph patterns with differential privacy. KDD '13, page 545–553. Association for Computing Machinery.

[Shen et al., 2021] Shen, Z., Xia, Z., and Yu, P. (2021). Pldp: Personalized local differential privacy for multidimensional data aggregation. *Security and Communication Networks*, 2021(1):6684179.

[Thakurta et al., 2017] Thakurta, A. G., Vyrros, A. H., Vaishampayan, U. S., Kapoor, G., Freudinger, J., Prakash, V. V., Legendre, A., and Duplinsky, S. (2017). Emoji frequency detection and deep link frequency. US Patent App. 15/640,266.

[Tokuda et al., 2013] Tokuda, K., Nankaku, Y., Toda, T., Zen, H., Yamagishi, J., and Oura, K. (2013). Speech synthesis based on hidden markov models. *Proceedings of the IEEE*, 101(5):1234–1252.

[Wang et al., 2023a] Wang, H., Zhang, Z., Wang, T., He, S., Backes, M., Chen, J., and Zhang, Y. (2023a). Privtrace: Differentially private trajectory synthesis by adaptive markov model. In *USENIX Security Symposium*.

[Wang and Kankanhalli, 2020] Wang, N. and Kankanhalli, M. S. (2020). Protecting sensitive place visits in privacy-preserving trajectory publishing. *Computers & Security*, 97:101949.

[Wang et al., 2023b] Wang, N., Wang, Y., Wang, Z., Nie, J., Wei, Z., Tang, P., Gu, Y., and Yu, G. (2023b). Privnud: Effective range query processing under local differential privacy. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 2660–2672. IEEE.

[Wang et al., 2018a] Wang, N., Xiao, X., Yang, Y., Hoang, T. D., Shin, H., Shin, J., and Yu, G. (2018a). Privtrie: Effective frequent term discovery under local differential privacy. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 821–832. IEEE.

[Wang et al., 2019a] Wang, N., Xiao, X., Yang, Y., Zhao, J., Hui, S. C., Shin, H., Shin, J., and Yu, G. (2019a). Collecting and analyzing multidimensional data with local differential privacy. In *IEEE 35th International Conference on Data Engineering (ICDE)*, pages 638–649. IEEE.

[Wang et al., 2017] Wang, T., Blocki, J., Li, N., and Jha, S. (2017). Locally differentially private protocols for frequency estimation. In *Proc. of the 26th USENIX Security Symposium*, pages 729–745.

[Wang et al., 2019b] Wang, T., Ding, B., Zhou, J., Hong, C., Huang, Z., Li, N., and Jha, S. (2019b). Answering multi-dimensional analytical queries under local differential privacy. In *Proceedings of the 2019 International Conference on Management of Data*, pages 159–176.

[Wang et al., 2018b] Wang, T., Li, N., and Jha, S. (2018b). Locally differentially private frequent itemset mining. In *IEEE Symposium on Security and Privacy (SP)*. IEEE.

[Wang et al., 2019c] Wang, T., Li, N., and Jha, S. (2019c). Locally differentially private heavy hitter identification. *IEEE Transactions on Dependable and Secure Computing*, 18(2):982–993.

[Wang et al., 2019d] Wang, T., Yang, X., Ren, X., Yu, W., and Yang, S. (2019d). Locally private high-dimensional crowdsourced data release based on copula functions. *IEEE Transactions on Services Computing*, 15(2):778–792.

[Wang and Cheng, 2022] Wang, Y. and Cheng, X. (2022). Prism: Prefix-sum based range queries processing method under local differential privacy. In *2022*

IEEE 38th International Conference on Data Engineering (ICDE), pages 433–445. IEEE.

[Xiao et al., 2017] Xiao, Y., Xiong, L., Zhang, S., and Cao, Y. (2017). Loclok: Location cloaking with differential privacy via hidden markov model. *Proc. VLDB Endow.*, 10(12):1901–1904.

[Xiong et al., 2016] Xiong, S., Sarwate, A. D., and Mandayam, N. B. (2016). Randomized requantization with local differential privacy. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2189–2193.

[Yang et al., 2020a] Yang, J., Wang, T., Li, N., Cheng, X., and Su, S. (2020a). Answering multi-dimensional range queries under local differential privacy. *Proceedings of the VLDB Endowment*, 14(3):378–390.

[Yang et al., 2020b] Yang, M., Lyu, L., Zhao, J., Zhu, T., and Lam, K.-Y. (2020b). Local differential privacy and its applications: A comprehensive survey. *arXiv preprint arXiv:2008.03686*.

[Yang et al., 1997] Yang, Y., Pedersen, J. O., et al. (1997). A comparative study on feature selection in text categorization. In *International Conference on Machine Learning (ICML)*, volume 97, page 35. Citeseer.

[Ye et al., 2021] Ye, Q., Hu, H., Li, N., Meng, X., Zheng, H., and Yan, H. (2021). Beyond value perturbation: Local differential privacy in the temporal setting. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pages 1–10.

[Zhang et al., 2024] Zhang, B., Wang, Z., Li, H., Lei, Z., Cheng, J., and Gao, S. (2024). Information gain-based multi-objective evolutionary algorithm for feature selection. *Information Sciences*, page 120901.

[Zhang et al., 2018] Zhang, Z., Wang, T., Li, N., He, S., and Chen, J. (2018). Calm: Consistent adaptive local marginal for marginal release under local differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 212–229. ACM.