Exploring noise addition mechanisms for private matrix completion: Introducing the Huber Mechanism

A Project Report

submitted by

R ADITHYA GOWTHAM

in partial fulfilment of the requirements
for the award of the degree of

of

BACHELOR OF TECHNOLOGY
&
MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2022

THESIS CERTIFICATE

This is to certify that the thesis titled **Exploring noise addition mechanisms for private matrix completion: Introducing the Huber Mechanism**, submitted by **R Adithya Gowtham**, to the Indian Institute of Technology, Madras, for the award of the degrees of **Bachelor of Technology and Master of Technology**, is a bonafide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Dr. Sheetal Kalyani
Research Guide
Professor
Dept. of Electrical Engineering
IIT Madras, 600 036

Place: Chennai

Date: June 17, 2022

ACKNOWLEDGEMENTS

I would like to express my profound gratitude to my project guide, Prof. Dr. Sheetal Kalyani, for providing me the opportunity to work on this project and for her invaluable guidance. Her unique perspective, farsightedness, and breadth of knowledge played a vital role in bringing this project to fruition.

I also owe a lot to Gokularam M. and Thulasi Tholeti for their precious assistance and apt counsel at times of uncertainty. I learned a lot from them, and I could not have completed this thesis in time without their help. I wish them both the very best in their future endeavors.

A huge thanks to my parents for their unwavering support, love, and care through thick and thin. Mere words cannot express my gratitude.

I would also like to thank my friends and schoolmates who made even the most monotonous of times enjoyable. Finally, my gratitude to anyone supportive of my work, however small the contribution may have been.

ABSTRACT

KEYWORDS: Differential Privacy ; Noise addition mechanism; Privacy Budget; Budget-Accuracy Trade-off; Low-Rank Matrix Completion.

Differential Privacy is a theoretical privacy framework introduced for privacy-preserving data analysis. It is predominantly used to mask user participation in applications where the risk of the leakage of sensitive information exists. This is achieved by adding selective noise proportional to the privacy budget to certain portions of the algorithm through a noise addition mechanism. Matrix Completion is one such application where there is a large risk of leaking user information through the model recommendations. Many different private extensions of matrix completion algorithms are being introduced to mitigate this problem. In this work, we aim to explore the different types of noise addition mechanisms in the context of the Low-Rank Matrix Completion problem. This work also introduces an alternative type of noise addition mechanism, which provides a conditionally better trade-off between the privacy budget and accuracy. This is because any noise addition mechanism introduces noise into the system, which negatively affects the accuracy or predictive power of the algorithm. To address this problem, a new optimization procedure tailored to the new noise addition procedure is introduced. Ideally, we would like to obtain more accurate results with less privacy budget, and the newly proposed mechanism and optimization procedure take a step in this direction. These claims are backed by experimental results collected using both synthetically generated and real datasets.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DP	Differential Privacy
ML	Machine Learning
LRMC	Low-Rank Matrix Completion
ALS	Alternating Least Squares
PDF	Probability Distribution Function
CDF	Cumulative Distribution Function
SVT	Singular Value Thresholding
SVD	Singular Value Decomposition
FTF	Fast Trifactorization
IRLS	Iteratively Re-weighted Least Squares
MLE	Maximum Likelihood Estimate
A-IRLS	Alternating-Iteratively Reweighted Least Squares

NOTATION

$\mathbf{M}$	Matrices are denoted by uppercase letters
v	Vectors denoted by lowercase letters
a_i	The i_{th} row of the matrix $\mathbf{A}$ (as a column vector)
$\widetilde{a}_j$	The j_{th} column of the matrix $\mathbf{A}$ (as a column vector)
$\|\|a\|\|_p$	The ℓ_p-norm of a vector a
$\|\|A\|\|_F$	The Frobenius norm of matrix $\mathbf{A}$
$\|\|A\|\|_*$	The nuclear norm of matrix $\mathbf{A}$
$[p]$	$\{1, 2, \ldots, p\}$
$\mathcal{L}(b)$	Laplace distribution with mean 0 and scale b
$\mathcal{N}(\sigma^2)$	Gaussian distribution with mean 0 and variance σ^2
$\mathcal{H}(\alpha)$	Huber distribution with mean 0 and transition parameter α
$u \otimes v$	The Kronecker Product of vectors u and v.
$(.)^1$	The Inverse Operator for matrices
$(.)^\top$	The Transpose Operator for matrices

CHAPTER 1

INTRODUCTION

1.1 Differential Privacy

Privacy has always been a major concern in the 21st century, both for end-users and huge corporations. Big Data breaches have become commonplace in recent years, forcing governments, organizations, companies, and the common man to give more emphasis to privacy. On the contrary, every major breakthrough in ML techniques involves model training and hyperparameter optimization which require copious amounts of data. Due to the ever-increasing demand for innovation in ML fueled by its explosive growth in the current times, and the shortage of good data sources, research institutions frequently share their datasets to meet their requirements. This data often contains confidential or sensitive information about participants, which when disclosed improperly can have adverse effects on their lives, by disrupting their social life and/or causing physical injury. These risks are detailed in Korolova (2010) and Calandrino *et al.* (2011).

Sensitive Data
Data Curators
Adversaries

Figure 1.1: Three issues with making data private solved by introducing DP. [1]

The development of privacy-preserving techniques like **Differential Privacy** has become a necessity, as these models play a vital role in mitigating the problem. Differential Privacy was introduced by Dwork *et al.* (2014*a*) as a theoretical privacy framework

[1]Image courtesy: A Brief Introduction to Differential Privacy, Aaruran Elamurugaiyan.

that can be incorporated into an algorithm to provide an extended private version of it [Dwork *et al.* (2006), Dwork *et al.* (2007), Dwork *et al.* (2014*b*)]. These extensions have been incorporated into widely used algorithms in leading fields like gradient descent, deep learning, clustering, localization, and data mining [Song *et al.* (2013), Abadi *et al.* (2016), Xia *et al.* (2020) and, Friedman and Schuster (2010)].

As we speak, increasingly many organizations are leveraging DP to protect their end-users' sensitive information, such as their real-time location, clinical information, and life's events. These real-world DP applications include [2]:

- U.S. Census Bureau, to show commuting patterns [Machanavajjhala *et al.* (2008)].
- Google's RAPPOR, for telemetry [Erlingsson *et al.* (2014)].
- Google, for sharing historical traffic statistics in 2015 [3].
- Apple used DP in iOS 10 to improve Siri in 2016 [4].
- Microsoft, for telemetry in Windows [Ding *et al.* (2017)].
- LinkedIn, for advertiser queries [Rogers *et al.* (2020)].

Differential Privacy is achieved by adding noise to the output to make sure that the existence or non-existence of any user does not affect the output in a significant manner. Two noise addition mechanisms exist for DP: 1) The Laplacian mechanism which preserves $(\epsilon, 0)$-privacy and 2) The Gaussian mechanism which preserves (ϵ, δ)-privacy. The Laplacian mechanism requires a lesser privacy budget than the Gaussian mechanism, but it also compromises accuracy due to the heavy-tailed nature of Laplacian noise. We aim to give a provide a middle ground to these extremes through the Huber Mechanism.

1.2 Matrix Completion

Originally made prominent by the Netflix Prize in 2006 [Bennett and Lanning (2007)], the problem of completing a matrix with incomplete entries to attain a low-rank structure, also known as Low-rank Matrix Completion is now an extremely active research area in machine learning [Rendle *et al.* (2019)], because of its applications in key fields

[2]List sourced from Wikipedia.

[3]Tackling Urban Mobility with Technology, Andrew Eland.

[4]How Apple personalizes Siri without hoovering up your data, Karen Hao.

like Recommender Systems Luo *et al.* (2014), Phase retrieval Candes *et al.* (2011) and Image restoration He *et al.* (2015). Many giants like Meta, Amazon, and Netflix actively leverage Matrix Completion to provide recommendations to their users. Matrix Completion also finds applications in Gene Expression [Kapur *et al.* (2016)] and Drug Response [Hao *et al.* (2019), Liu *et al.* (2020)].

Several approaches have been proposed to solve this problem in existing literature. Cai *et al.* (2008) solved the nuclear-norm minimization problem originally proposed by Candes and Recht in 2008 [Candes and Recht (2008)] using singular value thresholding, and Candes and Plan (2010) proved that the resultant matrix had the lowest rank. Jain *et al.* (2012) expressed the target matrix $\mathbf{X}$ in the bi-linear form as a product of two matrices $\mathbf{X} = \mathbf{U}.\mathbf{V}^\top$ and solved the problem by alternative minimization. Some other approaches made use of matrix decomposition techniques like SVD ([citeliu2013ftf, Lu *et al.* (2015)] and QR decomposition [Liu *et al.* (2018)] to solve the problem. A majority of these solutions utilize iterative procedures to converge to an optimal solution.

Many LRMC models are prone to leak user data through their recommendations [Calandrino *et al.* (2011)]. So, we require a privacy framework like DP to protect the user-rating data. There are many researchers are working on providing private extensions to low-rank matrix completion problems. Liu *et al.* (2015) utilized a connection of DP to Bayesian posterior sampling [Ahn *et al.* (2012)] via Stochastic Gradient Langevin Dynamics [Welling and Teh (2011)] to achieve private collaborative filtering. Jain *et al.* (2018) provided the first provably joint differentially algorithm based on the Frank-Wolfe method for Quadratic Programming Frank and Wolfe (1956). Chien *et al.* (2021) extended the Alternating Least Squares using the Gaussian addition mechanism and derived the privacy guarantees for the same. They used an amalgam of noise addition, clipping, and thresholding to achieve the more relaxed (ϵ, δ)-DP. characteristic of the Gaussian mechanism. This mechanism provides good accuracy, but compromises a lot on the privacy budget, especially if the data is more ℓ_1-sensitive.

1.3 Contributions of the thesis

The main contributions of this thesis are:

1. Introduced the novel Huber mechanism of noise addition and derived the privacy bounds to provide a better budget-accuracy trade-off for LRMC.
2. Explored the various kinds of noise addition mechanisms for preserving differential privacy in the context of Low-Rank Matrix Completion.
3. Introduced a new optimization procedure for LRMC problems tailored to additive Huber noise for better accuracy with less privacy budget.
4. Experimentally validated the claims made. Tabulated and interpreted the results obtained in the experiments.

1.4 Chapter-wise organization

In Chapter 2, the fundamentals of DP are described in a detailed manner, including the formal definition of Differential Privacy in Section 2.1, ℓ_1 and ℓ_2 sensitivity in Section 2.2 and the Gaussian and Laplacian noise addition mechanisms in Section 2.3.

Chapter 3 provides a detailed understanding of Matrix Completion, specifically Low-Rank Matrix Completion (LRMC). This includes formally posing the LRMC Problem, LRMC methods like Singular Value Thresholding, Alternating Least Squares (ALS) and Iteratively Reweighted Least Squares (IRLS) in Sections 3.1, 3.2, 3.4.1, and 3.5 respectively. We also discuss some applications of LRMC, like Collaborative Filtering and Social Network Recovery.

Chapter 4 introduces the new Huber mechanism of noise addition for differential privacy in Section 4.5. Some relevant topics like the PDF and CDF of Huber noise, sampling from the Huber noise PDF, and the derivations of the privacy bounds are explored in Sections 4.1, 4.2, and 4.4. Finally, we discuss a new optimization procedure tailored to Huber Noise in Section 4.6.

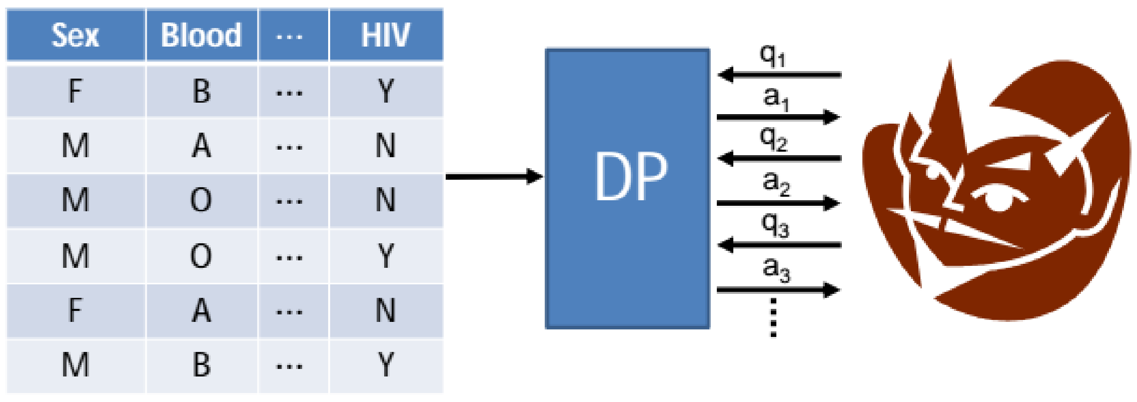
Chapter 5 details the privacy guarantees provided by the algorithms for different noise addition mechanisms in Section 5.1, and the experiments conducted for determining the accuracy of these algorithms in Section 5.2. These results are tabulated and interpreted to provide the conclusions.

Chapter 6 concludes the thesis by providing a summary of all the innovations introduced in the thesis and the results from the experiments section. It also discusses future work that can improve upon the existing theory. This includes topics like the exploration of the new noise addition mechanism in the context of other applications of differential privacy and introducing a more suitable definition of sensitivity.

CHAPTER 2

DIFFERENTIAL PRIVACY

Differential Privacy is a theoretical framework introduced by Dwork *et al.* (2014*a*) that enables the data-owner to publicly share or distribute a dataset while withholding the private sensitive information of individuals. DP allows for the expression of patterns in the dataset's clusters while withholding individual information. It can also be thought of as a strict, mathematical embodiment of privacy in the context of statistical or ML frameworks. The core idea behind DP is that if the effect of a random minor perturbation in the dataset is negligible enough, then the result of any malicious query cannot hold significant information about any particular individual, thus preserving privacy. Going by the mathematical definition of DP, privacy is ensured by enforcing a constraint on any procedure or algorithm distributing private information of end-users in the form of a statistical dataset or otherwise. Any procedure or algorithm that satisfies this constraint is termed as *differentially private*.



Sex	Blood	···	HIV
F	B	···	Y
M	A	···	N
M	O	···	N
M	O	···	Y
F	A	···	N
M	B	···	Y

Figure 2.1: DP preventing sensitive information from being leaked through the responses to the queries of the brown adversary. [1]

Differentially private algorithms are being increasingly leveraged by governmental institutions to disseminate large-scale demographic information or key statistical aggregates while still ensuring that the public responses to any surveys, like the census,

[1]Image Courtesy: Differential Privacy, Harvard University Private Tools Project.

remain confidential. Some companies like Linkedin [Rogers *et al.* (2020)] and Apple [2] are also leveraging DP to gather and store information about end-user behaviour while ensuring that this information remains obscure even to employees who utilize this data to train models.

Figure 2.2 details the kinds of information present in any dataset. General information is a property of the dataset as a whole and does not belong to any unique individual data subject. The other side of the coin is private information, which is attributed to one and only one individual.

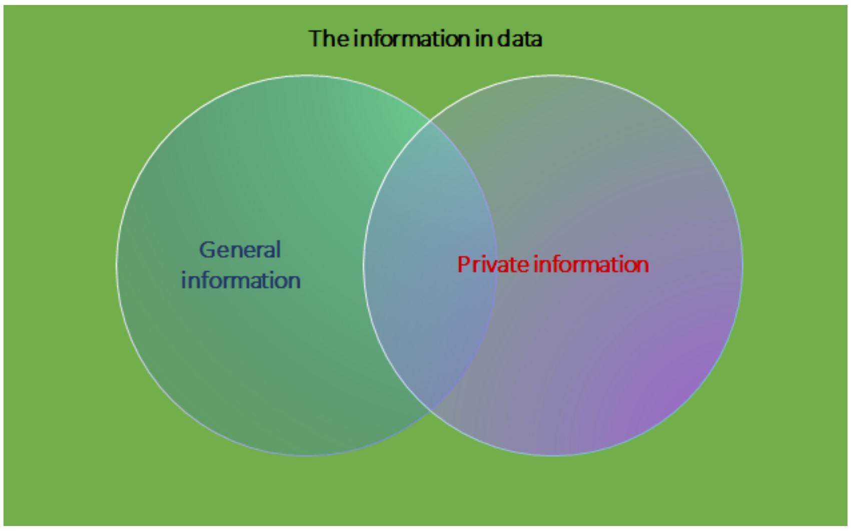


Figure 2.2: Types of information in Data as per DP. [3]

DP aims to address the paradox posed by privacy-preserving data analysis, i.e. to obtain information about a population without gaining any information about any particular individual. To put this in perspective, consider the following thought experiment. Consider a medical study using a medical dataset that proves alcohol increases the risk of heart attacks, which may heighten the estimate of an insurance company's prolonged medical costs of an alcoholic. So, does this mean the study has harmed the interest of the alcoholic? The impact is indeterminate: if insurance premiums rise, then they may be negatively impacted based on their social standing, or they may be positively impacted, as the study may convince them to give up drinking after learning of the hazards of alcohol consumption. All that is deterministic is that more of the alcoholic's information is public after the study than before. So does this mean their privacy is

[2]How Apple personalizes Siri without hovering up your data, Karen hao.
[3]Image Courtesy: Understanding Differential Privacy, An Nguyen.

compromised? The idea of Differential Privacy will maintain a view that the study did not harm the personal interest of the individual in any manner, as *the impact on the alcoholic is similar, irrespective of their participation* in the study. This means that it is the *inferences* of the study that impact them, not their individual participation.

Salient Features of DP:

- *Explicit Privacy budget*: A budget can be explicitly set for any algorithm or procedure that regulates the amount of privacy provided (by regulating the noise added to the algorithm's outputs) that trades off with its accuracy. It allows for easy comparison between different methods.
- *Aggregate Privacy*: DP allows for an overall budget and analysis for groups as a whole.
- *Free from Post-processing*: DP is immune to post-processing effects. An adversary cannot apply a procedure to the output of a private algorithm to gain any new private information without insider knowledge.
- *Easy to compose*: A well-defined privacy budget makes it straightforward to accumulate the privacy over multiple iterations. This allows for the building of complex DP frameworks from rudimentary components.

Let's go over some important definitions first before formally defining DP.

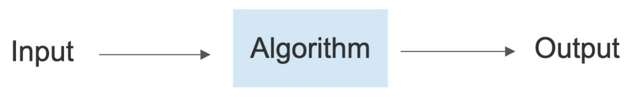
2.1 Formal defining Differential Privacy

A privacy mechanism $\mathcal{M}$ is a randomized algorithm that takes a database in matrix format as input and provides a response to a raised query. The idea of a randomized algorithm is explained in 2.1.1.

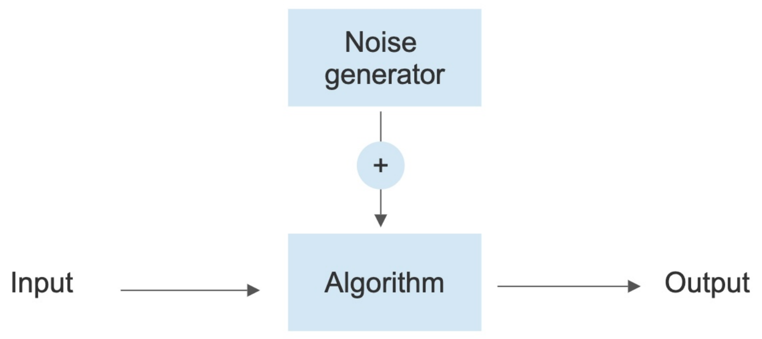
2.1.1 Randomized algorithms

DP works by introducing randomness into the output of an algorithm before public distribution. This is essential, as any *non-trivial* guarantee of privacy that needs to hold irrespective of any augmentation of information in the future requires a component of randomness, as perfectly deterministic quantities could always be exposed some day.

The working of randomized algorithms is illustrated by the flowcharts in Figure 2.3:



(a) A Deterministic Algorithm



(b) A Randomized Algorithm

Figure 2.3: Flowcharts of deterministic and randomized algorithms.

So it becomes essential to discuss the domain and range of random algorithms. Any randomized algorithm with domain X and discrete range Y is typically associated with a mapping from X to $P_{sim}(Y)$, the probability simplex over Y, defined as:

$$P_{sim}(Y) = \left\{ p \in \mathbb{R}^{|Y|} : p_i \geq 0 \; \forall i \text{ and } \sum_{i=1}^{|Y|} p_i = 1 \right\} \tag{2.1}$$

A randomized algorithm F with domain X and discrete range Y is denoted by the mapping $F : X \to P_{sim}(Y)$. For an input $x \in X$, the random algorithm F gives the output $F(x) = y_i$ with probability $p_i \; \forall \, y \in Y$. The probability space of the outputs is characteristic of the type of randomness the algorithm introduces into the input during processing.

2.1.2 Mathematical definition of Differential Privacy

The Differential privacy framework is defined mathematically as follows [Dwork *et al.* (2014*a*)]:

Definition 2.1.1 (Differential Privacy) *A randomized mechanism $\mathcal{M}$ preserves ϵ-differential privacy if for all datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ that differ on a single element (i.e. $\|\mathcal{D}_1 - \mathcal{D}_2\|_1 = 1$) and for all possible sets $\mathcal{A}$,*

$$\Pr(\mathcal{M}(\mathcal{D}_1) \in \mathcal{A}) \leq \exp(\epsilon) \cdot \Pr(\mathcal{M}(\mathcal{D}_2) \in \mathcal{A}). \tag{2.2}$$

where $\Pr(E)$ indicates the probability of occurrence of Event E. This idea is visually portrayed by the chart in 2.4:

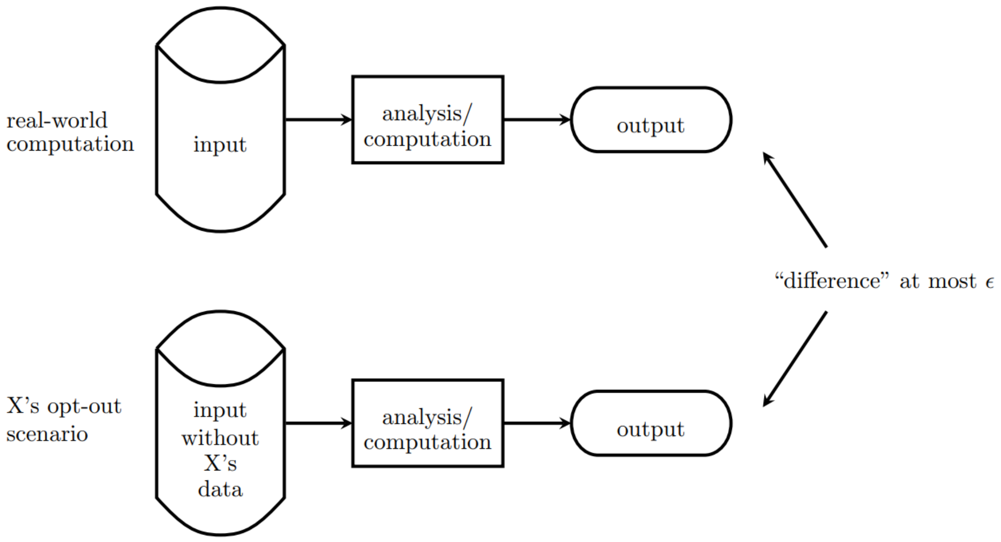


Figure 2.4: Illustration of the mathematical definition of DP. [4]

Such a mechanism $\mathcal{M}$ is said to be $(\epsilon, 0)$ differentially private. A weaker notation of privacy is defined for cases when ϵ differential privacy is preserved for the large part, excluding a small fraction δ:

Definition 2.1.2 *A randomized mechanism $\mathcal{M}$ is said to preserve (ϵ, δ)- differential privacy if for all datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ that differ on a single element and for all possible sets $\mathcal{A}$,*

$$\Pr(\mathcal{M}(\mathcal{D}_1) \in \mathcal{A}) \leq \exp(\epsilon) \cdot \Pr(\mathcal{M}(\mathcal{D}_2) \in \mathcal{A}) + \delta. \tag{2.3}$$

Such a mechanism $\mathcal{M}$ is said to be (ϵ, δ) differentially private.

[4]Image Courtesy: Understanding Differential Privacy, An Nguyen.

2.2 Sensitivity

The idea of sensitivity stems from the fact that every random algorithm is *sensitive* to noise, and such sensitivity works in our favour if it makes the algorithm behave in a privacy-preserving fashion [Dwork *et al.* (2006)]. To elucidate, the sensitivity of a random algorithm refers to the uncertainty in the response that we must introduce in order to hide the participation of a singular individual. In other words, it provides an upper bound on the "degree of perturbation" required to preserve privacy. Mathematically speaking, it captures the magnitude by which a single user's data can affect F in the worst case.

To determine an algorithm's sensitivity, the upper bound of the tentative perturbations in the output due to a fixed change in its input needs to be computed. So it goes without saying that the change introduced to the input needs to be a deterministic value for comparing the sensitivities of algorithms. But all our inputs are databases, so we need a rigorous definition of the distance between databases to proceed further.

Let us consider databases x as an unordered list of records from the superset of records $\mathcal{X}$. Without representing databases as a list, let us instead represent them using their histograms: $x \in \mathbb{N}^{|\mathcal{X}|}$ in which each entry x_i denotes the count of records in the database x of kind $i \in \mathcal{X}$. Here, $\mathbb{N}$ is the set of whole numbers.

The ℓ_1 norm of a database x is denoted as $\|x\|_1$ and defined as:

$$\|x\|_1 = \sum_{i=1}^{|\mathcal{X}|} |x_i|. \tag{2.4}$$

And the ℓ_1 distance of two databases x and y is defined as:

$$dist_{l_1}(x,\, y) = \|x - y\|_1 \tag{2.5}$$

It is worth noting here that $\|x\|_1$ is the number of records in database x, i.e. its size, and $\|x - y\|_1$ measures the number of records that *differ* between x and y. With the definitions in place, we move on to define sensitivity. The general mathematical definition of the ℓ_p-sensitivity of a function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ is given by:

$$\Delta f = \max_{\substack{x,\, y \in \mathbb{N}^{|\mathcal{X}|} \\ \|x-y\|_p = 1}} \|f(x) - f(y)\|_p \,. \tag{2.6}$$

where $\|.\|_p$ denotes the ℓ_p-norm. Note that there are only two predominant types of sensitivities in use, the ℓ_1 and ℓ_2 sensitivities.

2.3 Noise addition mechanisms

Noise addition mechanisms allow us to convert deterministic algorithms to randomized algorithms, and DP leverages this to introduce privacy into a non-private (usually) deterministic algorithm. There are two methods of noise addition to vector-valued functions that are widely used, the Laplace mechanism and the Gaussian mechanism.

2.3.1 The Laplace Mechanism

Functions of the form $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, also called numeric queries are the most rudimentary kind of database queries, as they straightforwardly map datasets to k real numbers. The Laplace mechanism draws noise from the Laplace distribution centered at 0 and having scale parameter b. The PDF is defined as:

$$\mathcal{L}(x;\, b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right). \tag{2.7}$$

This is visualized in Figure 2.5, where we can observe that the Laplacian distribution has a long tail probability, which is more apparent for higher variances. This makes the outputs of algorithms using the Laplacian mechanism outlier prone. Note that the variance of this distribution is $\sigma^2 = 2b^2$. Sampling from this distribution is denoted by $X \sim \mathcal{L}(0, b)$. This can be shortened to $\mathcal{L}(b)$ in this scenario as all additive noise in DP is zero mean. The latter notation is followed for the rest of the thesis. As the name suggests, the Laplace mechanism directly computes f, and perturbs each dimension with Laplacian noise.

Definition 2.3.1 (The Laplace Mechanism) *Given any vector-valued function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, the Laplace Mechanism is defined as:*

$$\mathcal{M}_{\mathcal{L}}(x, f(.), \epsilon) = f(x) + (L_1, L_2, ..., L_k) \tag{2.8}$$

where L_i are i.i.d random variables sampled from $\mathcal{L}(\Delta f_1/\epsilon)$, and Δf_1 is the ℓ_1-sensitivity of f.

Dwork *et al.* (2014*a*) proved that the Laplace mechanism preserves $(\epsilon, 0)$-DP.

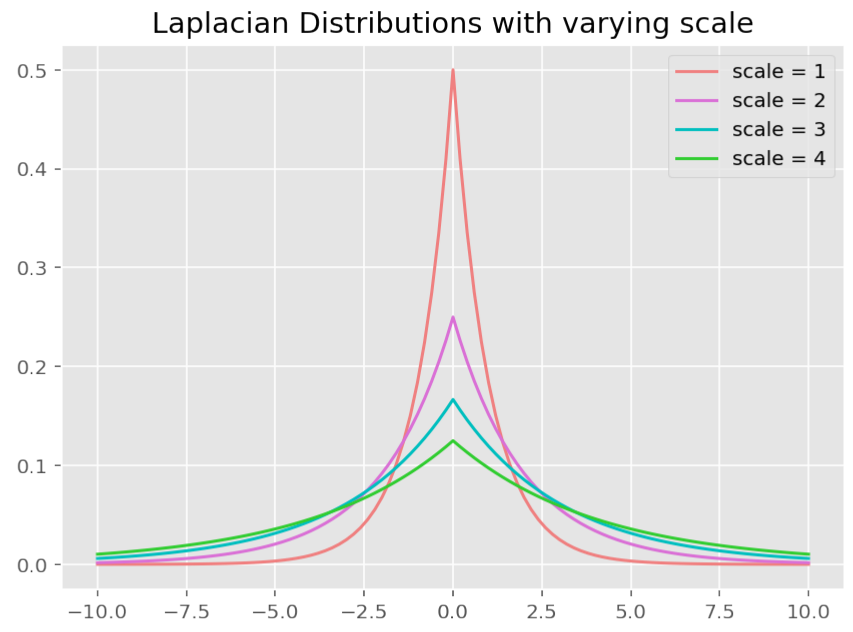


Figure 2.5: Zero-mean Laplacian Distributions with varying scale

The key advantage of the Laplace mechanism is that it adheres to the stronger $(\epsilon, 0)$ definition of DP. And, the drawbacks are:

- It is only really good for low sensitivity queries, usually w.r.t ℓ_1 sensitivity.
- Needs a generous privacy budget for a large number of queries, leading to less accurate results.
- Provides a simple solution to handle numeric queries, but cannot be applied to the non-numeric valued queries. But this drawback applies to all noise addition mechanisms available currently.

2.3.2 The Gaussian Mechanism

The Gaussian mechanism, suggestive of its name, adds Gaussian noise instead of Laplacian noise. The PDF of zero-mean Gaussian noise with variance σ^2 is given by:

$$\mathcal{N}(x;\, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{2.9}$$

This PDF is plotted in Figure 2.6 for better understanding. From the figure, we can understand that the Gaussian distribution is heavily centered around the mean, which allows the Gaussian mechanism to maintain good accuracy.

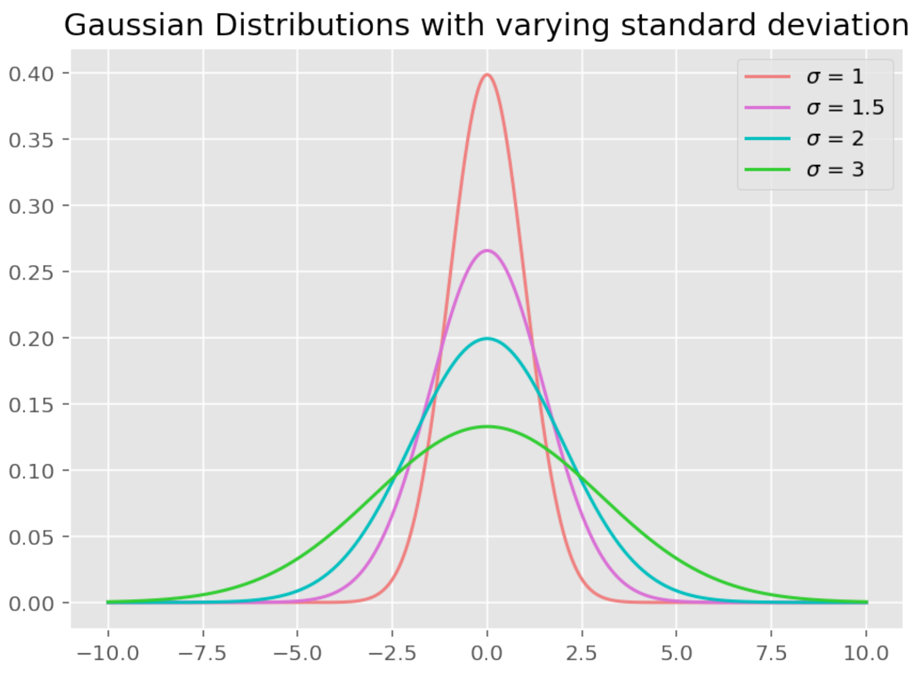


Figure 2.6: Zero-mean Gaussian Distributions with varying standard deviation

For a numerical query $f : \mathbb{N}^{|\mathcal{X}|} \rightarrow \mathbb{R}^k$, the Gaussian Mechanism is defined as:

$$\mathcal{M}_{\mathcal{N}}(x, f(.), , \epsilon, \delta) = f(x) + (N_1, N_2, ..., N_k) \tag{2.10}$$

where N_i are i.i.d random variables sampled from $\mathcal{N}(\sigma^2)$, where σ^2 takes the value:

$$\sigma^2 = \frac{2\Delta f_2^2 \log(1.25/\delta)}{\epsilon^2} \tag{2.11}$$

Here, Δf_2 is the ℓ_2 sensitivity of f. The Gaussian mechanism preserves the more relaxed (ϵ, δ)-DP [Dwork *et al.* (2014*b*)] .

The main advantage of the Gaussian mechanism over Laplacian is that for applications where the ℓ_2 sensitivity is much lower than ℓ_1 sensitivity, the Gaussian mechanism allows the addition of much less noise for the same budget. The drawback is that it only satisfies the relaxed (ϵ, δ)-DP definition, i.e. it ensures privacy only with probability $(1 - \delta)$, and it usually requires a greater privacy budget (as observed in 5.1).

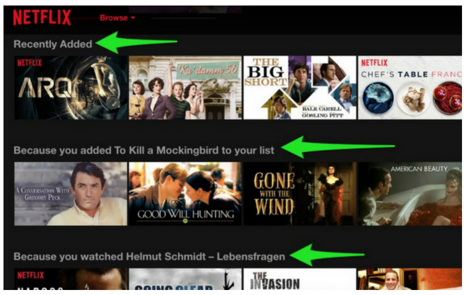
CHAPTER 3

MATRIX COMPLETION

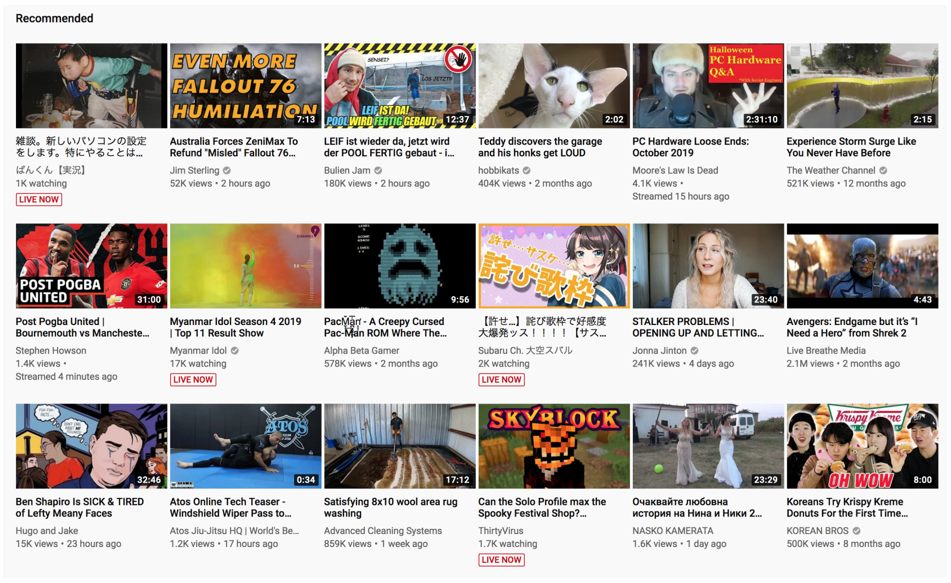
Matrix Completion, particularly Low-Rank Matrix Completion is an extremely active research area due to it being the key component in Recommender System based applications [Luo *et al.* (2014)]. Many notable giants like Amazon, Netflix, YouTube, and Meta make daily use of Matrix Completion-based algorithms to make appropriate recommendations for their user base, generating billions in revenue.

Frequently Bought Together
Price For All Three: $258.02
Add all three to Cart
Customers Who Bought This Item Also Bought
LOOK INSIDE!

(a) Amazon


NETFLIX
Browse
Recently Added
Because you added To Kill a Mockingbird to your list
Because you watched Helmut Schmidt – Lebensfragen

(b) Netflix



(c) Youtube

Figure 3.1: Widely used Recommendation Engines on the Internet. [1]

At its core, Matrix Completion is just filling in missing entries in an incomplete matrix, with only a small fraction of its entries filled as illustrated in Figure 3.2. One

[1]Image courtesy: Google images.

prominent example is the Netflix problem, where the rows of the matrix represent users, the columns of the matrix represent items, and a filled-in entry at (i, j) refers to the rating given by user-i to movie-j, if he/she has watched it, and is left unfilled otherwise. Therefore, completing the matrix indicates that the missing entries are known, which means we have estimated what rating user-i would have given movie-k, if he/she watches it. Based on these estimated ratings, we can simply recommend users highly-rated new movies that they have not watched yet.

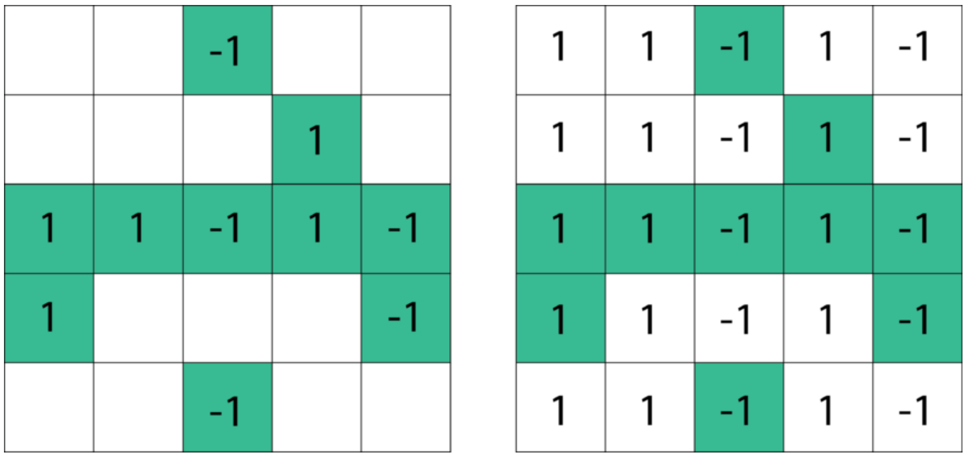


Figure 3.2: Matrix completion of an incomplete 5 $\times$ 5 matrix with a rank of 1. Left: The incomplete matrix; Right: The completed matrix. [2]

An apt and widely-used assumption is that the resulting completed ratings matrix would have a low-rank structure as in Figure 3.2, which itself is based on an underlying assumption that people watch movies based on their genres, like action, comedy, feel-good, or a mix of these genres. So we can safely say that the consideration is that the number of tastes people have in movies is limited to a small positive constant. Since similar people watch similar movies, the number of independent features in the completed matrix would be limited, resulting in a low-rank structure. This assumption can be generalized to any scenario where people rate items based on their preferences.

3.1 Posing the Low-Rank Matrix Completion Problem

The LRMC problem is posed in the following manner by Candes and Recht (2008): To find the matrix with the lowest rank $\mathbf{X}$ that matches the matrix $\mathbf{M}$ that we wish to

[2]Image Courtesy: Wikipedia.

complete for all observed entries, denoted by the set Ω:

$$\begin{aligned} \underset{\mathbf{X}}{\text{minimize}} \quad & rank(\mathbf{X}) \\ \text{subject to} \quad & \mathbf{X}_{ij} = \mathbf{M}_{ij} \;\; \forall (i,j) \in \Omega. \end{aligned} \tag{3.1}$$

This formulation is valid only when the entries are not corrupted by noise. Unfortunately, in the real world, the data we obtain is corrupted in most cases. For such data, the problem is formulated in the following fashion in Candes and Plan (2010) provided the rank of $\mathbf{X}$ is fixed to a pre-defined hyperparameter r:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \text{dist}(\mathbf{X}, M) \\ \text{subject to} \quad & \mathcal{P}_\Omega(\mathbf{X}) = \mathcal{P}_\Omega(\mathbf{M}), \\ & rank(\mathbf{X}) = r. \end{aligned} \tag{3.2}$$

where dist$(\mathbf{X}, \mathbf{M})$ is any distance measure for matrices, and $\mathcal{P}_\Omega(\mathbf{M})$ is defined as:

$$\mathcal{P}_\Omega(\mathbf{M})_{ij} = \begin{cases} \mathbf{M}_{ij}, & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

for any matrix $\mathbf{M}$, and 0 is set as the null value. The $\mathcal{P}_\Omega(.)$ constraint is also known as the equality constraint for matrix completion.

Unfortunately, the rank minimization problem is NP-Hard [Recht *et al.* (2008)], and cannot be solved easily. Therefore, many relaxations were proposed to the original problem in the interest of time complexity and computation resources. We will look at some of these approaches in the coming sections.

3.2 Singular Value Thresholding

Singular Value Thresholding [Cai *et al.* (2008)], or SVT is an algorithm that minimizes the nuclear norm of a matrix subject to the equality constraint for matrix completion.

Formally, the problem is defined as:

$$\begin{aligned} \underset{\mathbf{X}}{\text{minimize}} \quad & \tau \left\|\mathbf{X}\right\|_* + \frac{1}{2}\left\|\mathbf{X}\right\|_F^2, \\ \text{subject to} \quad & \mathcal{P}_\Omega(\mathbf{X}) = \mathcal{P}_\Omega(\mathbf{M}). \end{aligned} \tag{3.4}$$

The Frobenius norm of $\mathbf{X}$ is added to the objective to provide regularization. It has been conclusively shown by Cai *et al.* (2008) that solving this objective gives a good low-rank approximation to the incomplete matrix. But even this can't be solved directly in the current form since the equality constraint is non-convex. Taking the Lagrangian dual of the above optimization problem and rewriting the constraint, we get:

$$\mathbf{Y}^* = \underset{\mathbf{Y}}{\arg\min}\ \ f_\tau(\mathbf{X}) + \langle \mathbf{Y}, \mathcal{P}_\Omega(\mathbf{M} - \mathbf{X}) \rangle = \underset{\mathbf{Y}}{\arg\min}\ \ \tau \left\|\mathbf{X}\right\|_* + \left\|\mathbf{X} - \mathcal{P}_\Omega(\mathbf{Y})\right\|_F^2. \tag{3.5}$$

The global solution involves soft-thresholding over the singular values of optimal $\mathbf{Y}^*$. Despite its simplicity, SVT is an old technique that involves the computation of the Singular Value Decomposition, or SVD of the matrices to get results, which is computationally expensive. Recently, Matrix Factorization based approaches have picked up momentum due to their two key advantages over SVT, in being extremely fast and adequately accurate.

3.3 Matrix Factorization

Matrix Factorization is a collaborative filtering-based approach, which is based on the transitive principle that it is very likely for people who share an interest in one field to share common interests in many fields, which could be used to our advantage. Going by its roots, collaborative filtering is a technique for *filtering* out the preferences of a user by collecting the interest and tastes of many *collaborative* users. This method of recommending things is conveyed in Figure 3.3.

In Matrix Factorization-based approaches, the incomplete matrix is expressed as a product of two or three matrices. So, Matrix Factorization is also known as the Multiplicative Decomposition of matrices. After the decomposition, suitable procedures are applied over the individual matrices to get back a low-rank matrix closest to the origi-

[3]Image Courtesy: Collaborative Filtering in Pytorch, Neel Iyer.

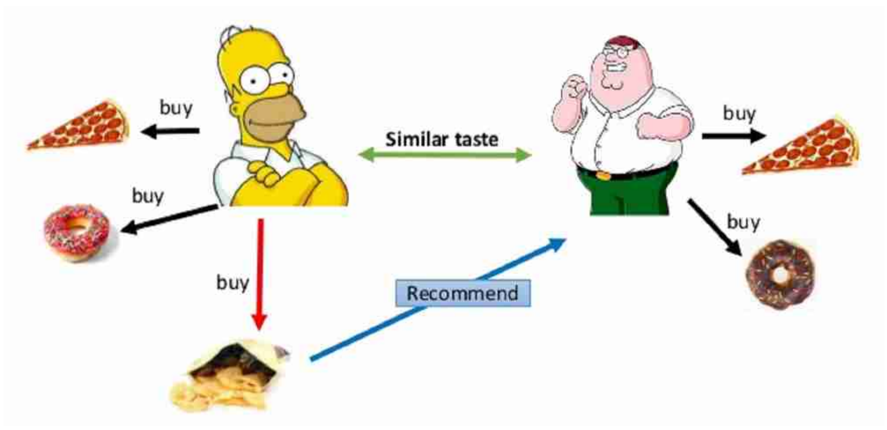


Figure 3.3: Making recommendations using user-based collaborative filtering. [3]

nal. Predominantly, two types of decompositions are used to factorize matrices. Let's look at them in more detail.

3.3.1 Singular Value Decomposition

SVD is the most widely known decomposition technique used to factorize matrices, due to its elegance and widespread utility. SVD decomposes the original $\mathbf{X}$ of shape $m \times n$ matrix into three matrices $\mathbf{U}$, Σ, and $\mathbf{V}$ of shape $m \times r$, $r \times r$, and $r \times n$ respectively (where r can take any value from 1 to the rank of $\mathbf{X}$), which when multiplied together give back the original matrix. Here, $\mathbf{U}$ is called the left singular matrix, Σ the singular value matrix and $\mathbf{V}^\top$ the right singular matrix. It is worth noting that Σ is a diagonal matrix consisting of the singular values of $\mathbf{X}$, which are the eigenvalues of $\mathbf{X}^\top\mathbf{X}$. Rows of $\mathbf{U}$ are made up of the left singular vector of $\mathbf{X}$, which are the eigenvector of $\mathbf{X}\mathbf{X}^\top$. Rows of $\mathbf{V}^\top$ consist of the right singular vectors of $\mathbf{X}$, the eigenvector of $\mathbf{X}^\top\mathbf{X}$.

Eckart and Young (1936) demonstrated that a rank-r approximation of the rank-r_0 matrix $\mathbf{X}$ is obtained by computing the SVD of the matrix, and trimming the last $r_0 - r$ rows of $\mathbf{U}$, the rows and columns of Σ, and the columns of $\mathbf{V}$ as depicted in Figure 3.4.

.

$$\underset{m\times n}{\overset{\hat{X}}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n}\\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn}\end{pmatrix}}} \approx \underset{m\times r}{\overset{U}{\begin{pmatrix} u_{11} & \cdots & u_{1r}\\ \vdots & \ddots & \\ u_{m1} & & u_{mr}\end{pmatrix}}} \underset{r\times r}{\overset{S}{\begin{pmatrix} s_{11} & 0 & \cdots\\ 0 & \ddots & \\ \vdots & & s_{rr}\end{pmatrix}}} \underset{r\times n}{\overset{V^\intercal}{\begin{pmatrix} v_{11} & \cdots & v_{1n}\\ \vdots & \ddots & \\ v_{r1} & & v_{rn}\end{pmatrix}}}$$

Figure 3.4: Low-rank approximation using SVD. [4]

3.3.2 Fast Trifactorization

Fast Trifactorization, or FTF was proposed by Liu *et al.* (2013) as a fast and accurate method to solve the Matrix Completion problem. In FTF, the rank r matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$ is factorized into three matrices:

$$\mathbf{X} = \mathbf{LDR} \tag{3.6}$$

where $\mathbf{L} \in \mathbb{R}^{m\times r}$, $\mathbf{D} \in \mathbb{R}^{r\times r}$, and $\mathbf{R} \in \mathbb{R}^{r\times n}$. Despite looking similar to SVD, there is no constraint on $\mathbf{D}$ to be a diagonal matrix in this case. Now, if $\mathbf{L}$ is a column orthogonal matrix and $\mathbf{R}$ is a row orthogonal matrix, then:

$$\|\mathbf{X}\|_* = \|\mathbf{D}\|_* \tag{3.7}$$

So, the nuclear norm minimization problem in SVT can be modified as:

$$\begin{aligned} \underset{\mathbf{D}}{\text{minimize}} \quad & \|\mathbf{D}\|_* \\ \text{subject to} \quad & \mathbf{L}^\top\mathbf{L} = I, \;\; \mathbf{R}\mathbf{R}^\top = I, \\ & \mathcal{P}_\Omega(L\mathbf{D}R) = \mathcal{P}_\Omega(M). \end{aligned} \tag{3.8}$$

Here, r becomes a preset hyperparameter that regulates the cost of FTF. Now, the variables $\mathbf{L}$, $\mathbf{D}$, and $\mathbf{R}$ are alternately optimized by fixing other variables. $\mathbf{L}$ and $\mathbf{R}$ are updated using QR decomposition, and $\mathbf{D}$ is updated by applying SVT to a matrix of size $r \times r$, which saves a huge amount of computation cost.

While there are tangible benefits to using this procedure, it is still reliant on SVD for recovering $\mathbf{D}$, which slows the procedure down. But, there was a need for an even faster procedure even if it was not as accurate, as speed matters much more than accuracy in Recommender System-based applications. The Alternating Least Squares algorithm proposed by Koren and Bell (2015) based on Bi-linear factorization addresses this issue.

3.3.3 Bi-linear factorization

The most straightforward and frequently used type of factorization in Collaborative Filtering is the Bi-Linear Factorization. The bi-linear form of $m \times n$ matrix $\mathbf{X}$ is given

[4] Image Courtesy: Matrix Factorization made easy, Rohan Naidu.

as:

$$\mathbf{X} = \mathbf{U} \cdot \mathbf{V}^\top \tag{3.9}$$

where $\mathbf{U}$ and $\mathbf{V}$ are of shape $m \times r$ and $n \times r$ respectively, where r is the matrix rank of $\mathbf{X}$. In collaborative filtering terminology, $\mathbf{X}$, $\mathbf{U}$, and $\mathbf{V}$ are referred to as the *Rating Matrix*, the *User matrix* the *Item matrix* respectively, as shown in Figure 3.5.

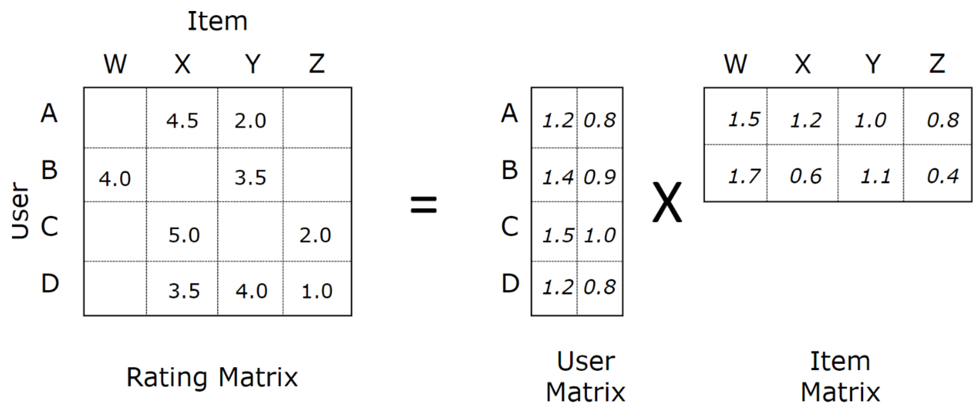


Figure 3.5: Bi-linear factorization of the Rating Matrix $\mathbf{X}$. [4]

The rows of $\mathbf{U}$ are called user-embeddings and the columns of $\mathbf{V}$ are called item-embeddings. Conventionally, the i^{th} row of $\mathbf{U}$ encodes the preference information of user-i, and the j^{th} row of $\mathbf{V}$ encodes the feature information of item-j. To obtain the complete matrix, we usually start with a random $\mathbf{U}$ and $\mathbf{V}$ of desired rank r and apply optimization techniques that utilize the similarity between users and items to get their multiplicative result as close to $\mathbf{X}$ as possible. The following section details the procedure to obtain the Bi-Linear form.

3.4 Alternating Minimization

This method, introduced by Jain *et al.* (2012) involves expressing the target rank **k** matrix **M** in bi-linear form as:

$$\mathbf{M} = \mathbf{U}\mathbf{V}^\top$$

where $\mathbf{U}$ and $\mathbf{V}$ are full-rank and $\mathbf{U} \in \mathbb{R}^{m\times k}$ is the user-embeddings matrix and $\mathbf{V} \in \mathbb{R}^{n\times k}$ is the item-embeddings matrix. $\mathbf{U}$ and $\mathbf{V}$ occupy lesser memory than $\mathbf{M}$ due to the low-rank assumption, saving us computation and memory costs. As an additional benefit, imposing constraints on users or items becomes easier when the rating matrix is expressed in this form. Note that this model assumes the incoherence of $\mathbf{M}$ which ensures that the singular vectors of $\mathbf{M}$ are not too "sparse" [Jain *et al.* (2012)].

The algorithm works by starting with randomly initialized $\mathbf{U}_0$ and $\mathbf{V}_0$ and updating them alternatively by fixing one and updating the other and vice-versa until convergence. The optimization formulation can be written as:

$$\hat{\mathbf{U}}, \hat{\mathbf{V}} = \underset{\mathbf{U}, \mathbf{V} \in \mathbb{R}^{n \times k}}{\text{argmin}} \left\| \mathcal{P}_\Omega \left(\mathbf{M} - \mathbf{U}\mathbf{V}^\top \right) \right\|_F^2 + \lambda \left\| \mathbf{U} \right\|_F^2 + \lambda \left\| \mathbf{V} \right\|_F^2$$

Note that regularization is included to maintain the incoherence condition.

3.4.1 Alternating Least Squares

When the loss function used is ℓ_2, the Alternating Minimization procedure is called Alternating Least Squares, or ALS [Koren and Bell (2015)]. This is one of the algorithms experimented with extensively in this thesis. This method has a closed-form solution for each time step $\mathbf{t}$ given by:

$$\forall i \quad \hat{\mathbf{U}}_i^{\mathbf{t}} = (\lambda \mathbf{I} + \sum_{j \in \Omega_i} \hat{\mathbf{V}}_j^{\mathbf{t}} \otimes \hat{\mathbf{V}}_j^{\mathbf{t}})^{-1} \sum_{j \in \Omega_i} \hat{\mathbf{M}}_{ij} \hat{\mathbf{V}}_j^{\mathbf{t}}, \tag{3.10}$$

$$\forall j \quad \hat{\mathbf{V}}_i^{\mathbf{t}+1} = (\lambda \mathbf{I} + \sum_{i \in \Omega_j} \hat{\mathbf{U}}_i^{\mathbf{t}} \otimes \hat{\mathbf{U}}_i^{\mathbf{t}})^{-1} \sum_{i \in \Omega_j} \left(\hat{\mathbf{M}}_{ij} \hat{\mathbf{U}}_i^{\mathbf{t}} + t_N \right) \tag{3.11}$$

where $u \otimes v$ refers to the outer product of the vectors u and v, $t_N \in \mathbb{R}^k$ is the noise vector sampled from any noise distribution, $\Omega \subseteq [n] \times [m]$, and for $i \in [n]$, $\Omega_i := j : (i, j) \in \Omega$ and for $j \in [m]$, $\Omega_j = i : (i, j) \in \Omega$. The addition of noise to the ALS procedure is done as shown in Equation 3.11, identical to the noise addition in Chien *et al.* (2021). This is because it was observed during experimentation that adding noise to the output during the ALS iterations directly leads to large fluctuations during the optimization and does not allow the algorithm to converge making the final outputs imprecise.

3.5 Iteratively Reweighted Least Squares

IRLS is a robust regression technique proposed by Holland and Welsch (1977) designed to find an M-estimator that obtains the MLE of a generalized linear model, to address the corruption of the normally distributed data by outliers from a different noise distribution. It has been proved in Dollinger and Staudte (1991) and Kalyani and Giridhar (2007) that the IRLS procedure approaches the MLE for Huber loss conditionally, making the most suitable procedure for optimizing procedures with additive Huber Noise (Section 4.6). Consider the estimation problem:

$$y = \mathbf{A}h + n \tag{3.12}$$

where $y \in \mathbb{R}^{n\times 1}$ is the target vector, $\mathbf{A} \in \mathbb{R}^{n\times m}$ is the feature matrix, $h \in \mathbb{R}^{m\times 1}$ is the vector of true parameters, and $n \in \mathbb{R}^{n\times 1}$ is the noise vector. According to the IRLS procedure, we first start with a random estimate $\hat{h}_0$. Then, the vector of residuals $r \in \mathbb{R}^{n\times 1}$ is computed at each time step $\mathbf{t}$, where $\mathbf{t} \geq 0$ as:

$$r_{\mathbf{t}} = y - \mathbf{A}\hat{h}_{\mathbf{t}}. \tag{3.13}$$

The LS criterion is now replaced as:

$$\hat{h}_{\mathbf{t}+1} = \underset{h}{\text{argmin}} \sum_{i=1}^{N} \rho\left(\frac{r_{\mathbf{t}}}{\sigma}\right) \tag{3.14}$$

where σ is the standard deviation of the noise and $\rho(.)$ is the objective function. Note that it should be continuous and convex. If the corrupting noise (for outliers) is drawn from the Laplace distribution, the objective function becomes the Huber loss [Huber (1964)]:

$$H_\alpha(x) = \begin{cases} x^2/2, & |x| \leq \alpha, \\ \alpha\left(|x| - \alpha/2\right), & |x| > \alpha. \end{cases} \tag{3.15}$$

The influence function for Huber loss is given as:

$$\phi(x) = \frac{\mathrm{d}(H_\alpha(x))}{\mathrm{d}x} = \begin{cases} x, & |x| \leq \alpha, \\ \alpha \cdot \text{sgn}(x), & |x| > \alpha. \end{cases} \tag{3.16}$$

where $sgn(.)$ is the signum function. Solving Equation 3.15 is equivalent to solving the equation for $\hat{h}_{\mathbf{t}}$:

$$\mathbf{A}^\top \mathbf{W} r_{\mathbf{t}} = \mathbf{0} \tag{3.17}$$

where the weight matrix $\mathbf{W_t} \in \mathbb{R}^{n \times n}$ is defined as:

$$\mathbf{W_t} = \operatorname{diag}\left\{\frac{\phi(r_\mathbf{t}/\sigma)}{r_\mathbf{t}/\sigma}\right\} \tag{3.18}$$

and $\mathbf{0}$ is an $m \times 1$ vector of zeroes. This process is run iteratively for T timesteps to get the final IRLS estimate $\hat{h}_T$. While IRLS may not be a matrix completion technique, it will be used in the next chapter to introduce an optimization procedure for matrix completion tailored to the Huber mechanism.

CHAPTER 4

THE HUBER MECHANISM

As we saw in Section 2.3, Gaussian or Laplacian noise is added to confidential data to preserve DP through the Gaussian and Laplacian mechanisms respectively. We also saw that the Laplacian mechanism preserves $(\epsilon, 0)$-DP while the Gaussian preserves (ϵ, δ)-DP. The selection of a noise addition mechanism is done based on the particular requirements of the algorithm or procedure, mainly its sensitivity to the noise added.

But a trade-off always exists between privacy budget and model accuracy. For better accuracy, we need data that contains less noise, but preserving privacy requires the addition of more noise to mask individual participation. These two operations are contradictory in nature, hence the trade-off. Coming to noise addition, the Laplacian mechanism is more suited for applications where the privacy budget is constrained, whereas the Gaussian mechanism is more suited for applications where the data is more normally distributed since the noise just acts as part of the data. In other words, the Gaussian mechanism is more suitable for applications that have a *significantly* lower ℓ_2-sensitivity than ℓ_1-sensitivity since it allows for the addition of a smaller amount of noise to achieve the same privacy.

4.1 Huber Noise

An alternative to these types of noise addition mechanisms is proposed in this thesis. It is based on the Huber Loss function proposed by Huber (1964). The expression for Huber loss has already been given in Equation 3.15.

The PDF of *Huber Noise* is given as [Huber (1964)]:

$$p_X(x) = \mathcal{H}(\alpha) = N_\alpha \cdot \exp(-H_\alpha(x)) \tag{4.1}$$

where N_α is the normalization constant of the PDF, the expression for which is given as:

$$N_\alpha = \frac{1}{\sqrt{2\pi} \cdot \mathrm{erf}\left(\frac{\alpha}{\sqrt{2}}\right) + \frac{2}{\alpha} \cdot \exp\left(-\frac{\alpha^2}{2}\right)} \tag{4.2}$$

Here, erf(.) is the Gauss error function, defined as:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \mathrm{d}t$$

Huber noise is symmetrically distributed such that the PDF takes on the form of the Gaussian distribution in the central portion, also called the head, and the Laplacian distribution in the tail. Gaussian, Laplacian, and Huber distributions of the same variance are plotted in Figure 4.1 for a better perspective:

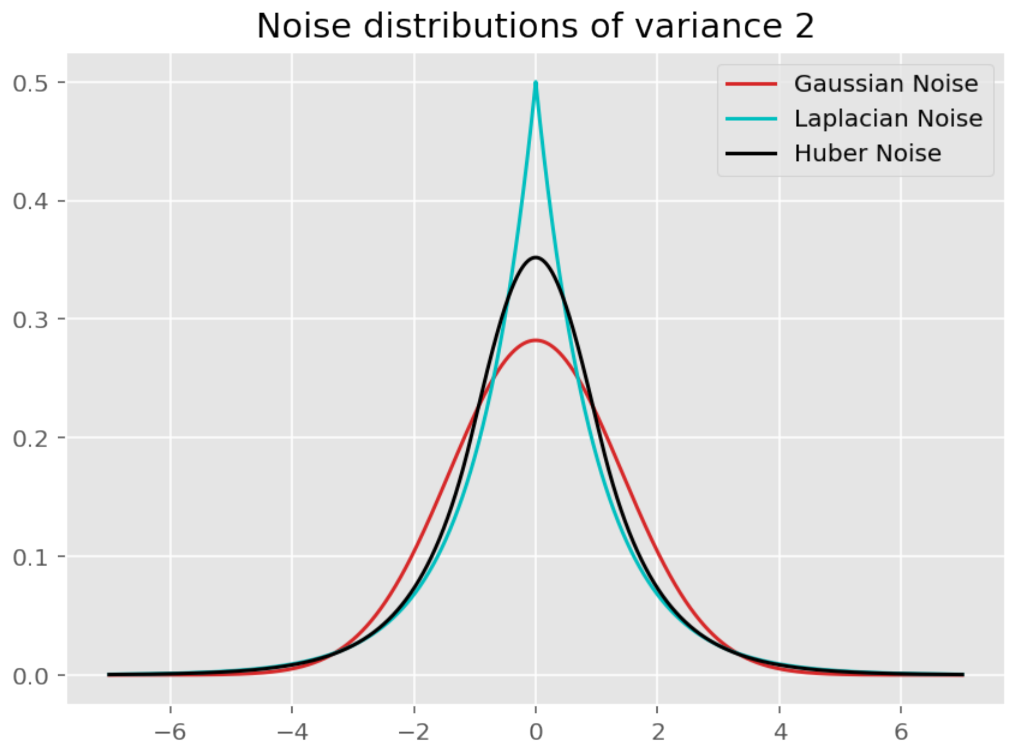


Figure 4.1: The plot of the noise PDFs with variance 2

The transition parameter α is altered to control the mix between the Gaussian and Laplacian distributions. We can set the transition parameter to the desired value so that the PDF matches the shape we require. This is illustrated in Figure 4.2

As seen in Figure 4.1, the tail of the PDF is similar to the Laplacian PDF. Hence, it is expected that the Huber noise addition mechanism similarly achieves $(\epsilon, 0)$-privacy. The detailed proof is given in Section 4.4. At the same time, the center of the Huber PDF takes the shape of the Gaussian distribution. So, for larger values of the transi-

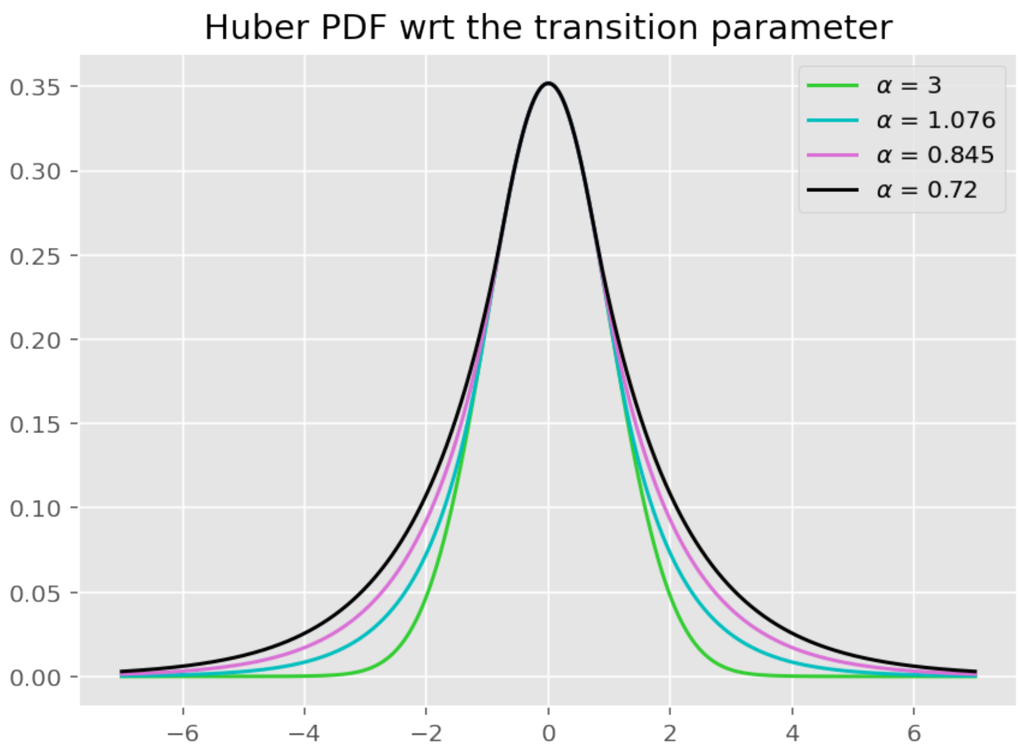


Figure 4.2: The variation in the shape of the Huber PDF with α

tion parameter, the sampled noise will mimic the behaviour of samples drawn from the Gaussian distribution, which makes the Huber mechanism more similar to the Gaussian mechanism. So this allows the Huber mechanism to add a smaller amount of noise to achieve stronger privacy guarantees like the Laplacian distribution.

4.2 Sampling from the Huber CDF

Before deriving the privacy bounds for the Huber Mechanism, it is important to know how to sample from the Huber Distribution. We need to use the Inverse-CDF method to draw samples from the Huber PDF because it is not possible to directly sample from the Huber PDF in code and, the key ingredient for the Inverse-CDF technique is the inverse function of the Huber CDF, which we explore in the following subsection.

4.2.1 The Huber CDF and Inverse-CDF

The complete derivations for the CDF and Inverse-CDF of Huber noise are given in Appendix B. The Huber CDF is as follows:

$$F_X(x) = \begin{cases} \frac{1}{\alpha \cdot N(\alpha)} \cdot \left(e^{\alpha(x+\alpha/2)}\right), & ;\, x \leq -\alpha \\ \frac{1}{\alpha \cdot N(\alpha)} \cdot e^{-\alpha^2/2} + \frac{1}{N(\alpha)} \cdot \sqrt{\frac{\pi}{2}} \cdot \left(\text{erf}\left(\frac{x}{\sqrt{2}}\right) - \text{erf}\left(\frac{-\alpha}{\sqrt{2}}\right)\right), & ;\, -\alpha \leq x \leq \alpha \\ \frac{1}{\alpha \cdot N(\alpha)} \cdot \left(2e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\alpha/\sqrt{2}\right) - e^{-\alpha(x-\alpha/2)}\right), & ;\, x > \alpha \end{cases} \tag{4.3}$$

And the Inverse CDF for Mixed-Huber Noise is given as:

$$F_Y^{-1}(y) = \begin{cases} \frac{1}{\alpha} \cdot \log\left(\alpha \cdot N(\alpha) \cdot y\right) - \frac{\alpha}{2} & ;\, y \in (0,\, y_1) \\ \sqrt{2} \cdot \text{inv\_erf}\left(\left(y \cdot N(\alpha) - \frac{1}{\alpha} \cdot e^{-\alpha^2/2}\right) \cdot \sqrt{\frac{2}{\pi}} + \text{erf}\left(\frac{-\alpha}{\sqrt{2}}\right)\right) & ;\, y \in [y_1,\, y_2] \\ \frac{1}{\alpha} \cdot \log\left(2e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right) - y \cdot \alpha \cdot N(\alpha)\right) + \frac{\alpha}{2} & ;\, y \in (y_2,\, 1) \end{cases} \tag{4.4}$$

where inv_erf(.) is the inverse error function. The expressions for y_1 and y_2 are given in Equations B.23 and B.24 respectively.

4.2.2 Sampling using the Inverse-CDF method

In the Inverse-CDF method of sampling, samples are drawn from the uniform distribution first. These samples are then passed on to the Inverse CDF of Huber noise $F_Y^{-1}(y)$ given in 4.4. The output of the Inverse CDF function is the required sample of Huber noise. To verify if the derivations in Appendix B are right, 1000000 samples are drawn from the Huber PDF and a histogram of the samples is plotted in Figure 4.3.

4.3 Relationship between α and variance

As observed in Figure 4.2, the variance of the Huber PDF changes when α is changed, and so a proper variance derivation is essential to understand the behaviour of Huber

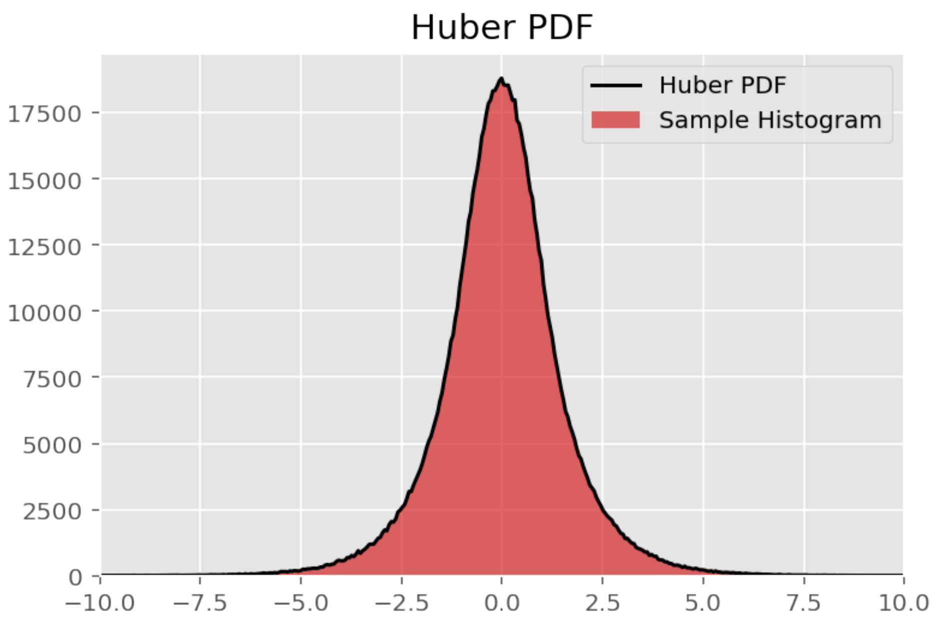


Figure 4.3: The histogram of samples.

noise. The noise is zero-mean, i.e. $E(X) = 0$. Thus, the variance is given by

$$\sigma^2 = E(X^2) - \big(E(X)^2\big)$$

so we just need to calculate $E(X^2)$:

$$E(X^2) = \int_{-\infty}^{\infty} x^2 \cdot p_X(x) \cdot dx$$

$$\begin{aligned} E(X^2) &= \frac{1}{N(\alpha)} \left(\int_{-\infty}^{-\alpha} x^2 e^{\alpha(x+\alpha/2)}\, dx \; + \; \int_{-\alpha}^{\alpha} x^2 e^{-x^2/2}\, dx \; + \; \int_{\alpha}^{\infty} x^2 e^{-\alpha(x-\alpha/2)}\, dx \right) \\ &= \qquad\qquad I_1 \qquad\qquad + \qquad I_2 \qquad + \qquad\qquad I_3 \end{aligned}$$

Since our noise PDF is an even function, and so is $f(x) = x^2$:

$$\begin{aligned} I_1 &= I_3 \\ I_2 &= \frac{1}{N(\alpha)} \cdot \left(\sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right) - 2\alpha e^{-\alpha^2/2} \right) \\ I_3 &= \frac{1}{N(\alpha)} \cdot \left(\int_{\alpha}^{\infty} x^2 \cdot e^{-\alpha(x-\alpha/2)} \cdot dx \right) \end{aligned}$$

$$= \frac{1}{N(\alpha)} \cdot \left(e^{\alpha^2/2} \cdot \left(\frac{2 \cdot (\alpha^2+1) \cdot e^{-\alpha^2}}{\alpha^3} + \alpha \cdot e^{-\alpha^2} \right) \right)$$

$$\implies I_3 = \frac{1}{N(\alpha)} \cdot \left(\frac{2 \cdot (\alpha^2+1) \cdot e^{-\alpha^2/2}}{\alpha^3} + \alpha \cdot e^{-\alpha^2/2} \right)$$

Summing up the individual integrals, we get:

$$\sigma^2 = E(X^2) - E(X)^2 = I_1 + I_2 + I_3 + 0 = 2I_3 + I_2$$

$$\sigma^2 = \frac{1}{N(\alpha)} \cdot \left(\frac{4}{\alpha^3} \cdot (\alpha^2+1) \cdot e^{-\alpha^2/2} + \sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right) \right)$$

$$\sigma^2 = \frac{\frac{4}{\alpha^3} \cdot (\alpha^2+1) \cdot e^{-\alpha^2/2} + \sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right)}{\frac{2e^{-\alpha^2/2}}{\alpha} + \sqrt{2\pi} \cdot = \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right)}$$

$$\implies \sigma^2 = \frac{4 \cdot (1 + \frac{1}{\alpha^2}) \cdot e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right)}{2e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right)} \tag{4.5}$$

This function is plotted in Figure 4.4, where we can observe that the relationship between α and variance decays extremely fast, starting from close to ∞ at $\alpha \to 0$, to almost 1 at just $\alpha = 3$. The function is not defined for $alpha < 0$, and it asymptotically approaches ∞ at 0 implying that it is not defined at 0, so the domain of the function is $(0, \infty)$. We can also observe that the variance reaches 1 asymptotically as $\alpha \to \infty$, implying that the range of the function is $(1, \infty)$. For $\alpha \geq 3$, the behaviour of the Huber distribution is very similar to that of the Gaussian distribution, and for $\alpha \leq 0.7$, its behaviour is practically identical to the Laplacian distribution.

4.4 Deriving the Privacy bounds

In this section, we formally derive the privacy guarantees of the Huber mechanism. **Proposition:** The Huber mechanism guarantees $(\epsilon, 0)$ differential privacy with $\epsilon = \alpha \cdot \Delta f_1$, where Δf_1 is the ℓ_1- sensitivity of the model.

Proof: Given that Δf_1 is the sensitivity of the model, we determine the ratio of Huber probabilities between a given point x and $x + \Delta f_1$. As Δf_1 indicates the maximum

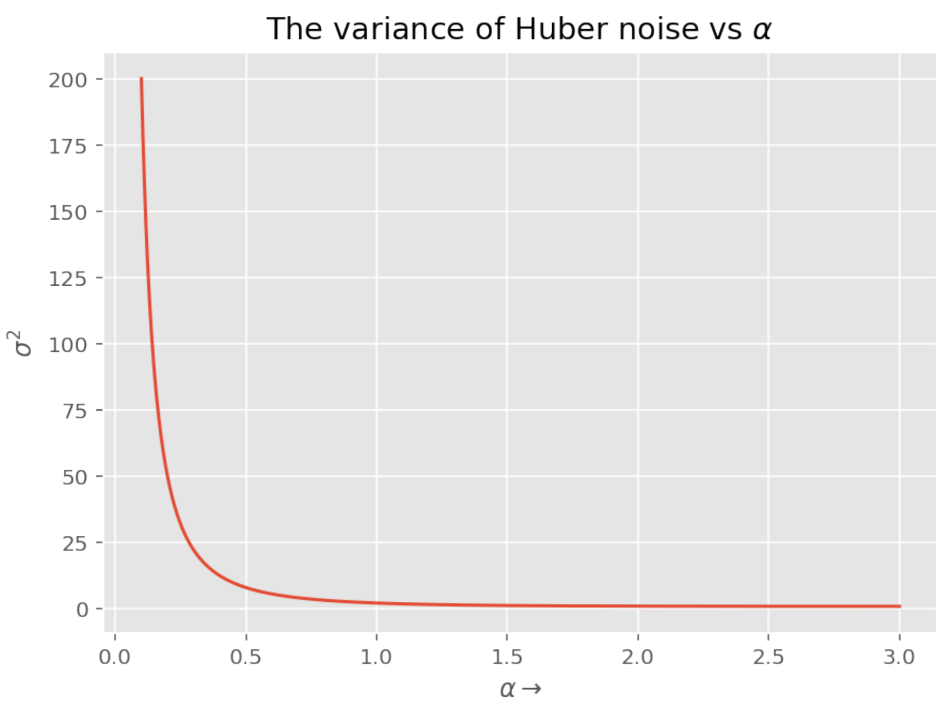


Figure 4.4: Variance vs alpha

distance between two data points in a model, this ratio is indicative of ϵ according to Definition 2.2. Note that this is also similar to the ratio considered to provide differential privacy guarantees for the Laplace mechanism in Dwork *et al.* (2014*a*).

$$\begin{aligned}\frac{p(x;\alpha)}{p(x+\Delta f_1;\alpha)} &= \frac{N_\alpha \exp(-H_\alpha(x))}{N_\alpha \exp(-H_\alpha(x+\Delta f))} \\ &= \exp(H_\alpha(x+\Delta f_1) - H_\alpha(x))\end{aligned}$$

Define $g(x)$ as

$$g(x) = H_\alpha(x+\Delta f_1) - H_\alpha(x)$$

It is evident from the nature of the two functions $H_\alpha(x+\Delta f_1)$ and $H_\alpha(x)$ that $g(x)$ is piece-wise defined. We consider two cases based on the behaviour of the function: $\Delta f_1 \leq 2\alpha$ and $\Delta f_1 > 2\alpha$. We derive the bounds for the two cases separately in Appendix A and verify that the upper bound for $g(x)$ in both cases is given by $\alpha \cdot \Delta f_1$, **thus validating our earlier proposition**.

4.5 Formally defining the Huber Mechanism

The Huber mechanism directly computes f, and perturbs each dimension with noise drawn from the Huber Distribution. Given any vector-valued function $f : \mathbb{N}^{|\mathcal{X}|} \rightarrow \mathbb{R}^k$, the Huber Mechanism is defined as:

$$\mathcal{M}_{\mathcal{H}}(x, f(.), \epsilon) = f(x) + (H_1, H_2, ..., H_k) \tag{4.6}$$

where H_i are i.i.d random variables sampled from $\mathcal{H}(\epsilon/\Delta f)$, and Δf is the ℓ_1-sensitivity of f. The Huber mechanism preserves $(\epsilon, 0)$-DP.

4.6 The Alternating-IRLS procedure

Since the ALS procedure is more suited for a normal distribution of the noise, we need a more suitable optimization algorithm for Huber noise. This issue is addressed by the Alternating-IRLS, or A-IRLS procedure for matrix completion 1. This procedure is based on the Alternating Minimization procedure given in Jain *et al.* (2012). The noise addition is in the same location as Chien *et al.* (2021) and Jain *et al.* (2018), because it was observed that adding noise to the output of each iteration introduced a lot of fluctuation in the optimization procedure while performing the experiments in 5. . This did not allow the algorithm to converge and thus produced imprecise results.

Since the noise is only added to the item embeddings, the user embeddings are optimized using ALS which saves time. The item embeddings are optimized using the procedure given in 3.5 as Huber noise is being added. Note that the optimization computation for each item embedding is done privately, which has two benefits: 1) This can be parallelized to save time and 2) We can make sure that the computation is done securely to avoid leakage to adversaries.

After these alternating steps are done, the final completed matrix is obtained by multiplying the user and item embeddings together.
To add a different type of noise, like Laplacian or Gaussian, we just replace the noise generation procedure in Line 11 of Algorithm 1 with the corresponding procedure of the noise type.

Algorithm 1: A-IRLS

```
Input: Incomplete data matrix $\mathbf{X}$, assumed rank $r$, regularization parameter $\lambda$,
       privacy budget $\epsilon$, sensitivity $\Delta f_1$, number of iterations $T$, number of
       IRLS iterations $N$.
Output: The completed matrix $\widehat{\mathbf{M}}$

Initialize $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ with random entries
for $T$ iterations do
    for $i \in \mathbb{N}_m$ do
        $\widehat{u}_i \leftarrow \left( (\widehat{\mathbf{V}}_{\Omega_i, \mathbb{N}_r})^\top \widehat{\mathbf{V}}_{\Omega_i, \mathbb{N}_r} + \lambda \mathbf{I}_r \right)^{-1} (\widehat{\mathbf{V}}_{\Omega_i, \mathbb{N}_r})^\top \mathcal{P}_{\Omega_i}(x_i)$
    end
    for $j \in \mathbb{N}_n$ do
        Randomly initialize $\widehat{\theta}_0 \sim \mathcal{U}(0,1)^r$
        for $k \in N$ do
            res $\leftarrow \mathcal{P}_{\widetilde{\Omega}_j}(\widetilde{\mathbf{x}}_j) - \widehat{\mathbf{U}}_{\widetilde{\Omega}_j, \mathbb{N}_r} \widehat{\theta}_{k-1}$
            $W_k \leftarrow$ diag($\psi$(res) / res)
            Generate Huber noise vector $t_H \sim \mathcal{H}(\epsilon / \Delta f_1)^r$
            $A \leftarrow (\widehat{\mathbf{U}}_{\widetilde{\Omega}_j, \mathbb{N}_r})^\top W_k \widehat{\mathbf{U}}_{\widetilde{\Omega}_j, \mathbb{N}_r} + \lambda \mathbf{I}_r$
            $\widehat{\theta}_k \leftarrow \left( A \right)^{-1} \left( (\widehat{\mathbf{U}}_{\widetilde{\Omega}_j, \mathbb{N}_r})^\top W_k \mathcal{P}_{\widetilde{\Omega}_j}(\widetilde{x}_j) + t_H \right)$
        end
        $\widehat{v}_j \leftarrow \widehat{\theta}_N$
    end
end
return $m_i = \mathbf{V} \widehat{u}_i, \ i = 1, 2, \ldots, m$
```

CHAPTER 5

EXPERIMENTS AND RESULTS

In this chapter, we present extensive simulation results on the privacy and accuracy of the ALS and A-IRLS algorithms with all three types of noise addition mechanisms for LRMC. We detail all the observations obtained from the simulation results and provide the inferences gained.

5.1 Privacy

In this section, we take a closer look at the privacy budget of all three noise addition mechanisms when noise of similar variance is added. This is a convenient way to compare the privacy provided by the noise addition mechanisms, since a smaller privacy budget for smaller variance implies that we have budget to spare for a better privacy guarantee without affecting model accuracy.

Note that it is assumed that the sensitivity of all algorithms $\Delta f_1 = \Delta f_2 = \Delta f$, where Δf is set to the maximum possible value, the range of all the datasets considered for the experiments, which is 5. This is not always the case but is essential to get a fair and objective sense of the privacy budget requirements of the various mechanisms. Therefore, the sensitivity Δf is set to 5 and the numerical evaluation of the theoretical budget required for similar variance settings is tabulated in Table 5.1:

Table 5.1: Budgets of the various mechanisms for a set noise variance.

Variance	**1**	**2**	**3**	**4**
Gaussian	$(15.96, 10^{-5})$	$(11.29, 10^{-5})$	$(9.22, 10^{-5})$	$(7.98, 10^{-5})$
Laplacian	$(7.07, 0)$	$(5.00, 0)$	$(4.08, 0)$	$(3.54, 0)$
Huber	$(15.00, 0)$	$(5.38, 0)$	$(4.23, 0)$	$(3.60, 0)$

Please note that the Huber noise variance approaches 1 as $\alpha \to \infty$ asymptotically, but it decays fast and is practically equal to 1 at $\alpha = 3$. So, we are using this value whenever the noise is of unit variance.

Key takeaways from Table 5.1:

- Laplacian noise has the lowest privacy budget of all three algorithms, but Huber is close when the variance is ≥ 2.
- Gaussian noise has the highest privacy budget, especially when the ℓ_2 sensitivity is close to the ℓ_1 sensitivity, despite the δ-relaxation of DP. It also doesn't decrease as much with increasing variance. For applications with a stricter budget, we might need to add a lot of noise due to which the accuracy may take a big hit.
- The behaviour of Huber noise is similar to Gaussian when the variance is low, where both have high privacy budgets.
- For high variance noise addition, Huber noise behaves like Laplacian noise and has a low privacy budget.

5.2 Accuracy

In this section, we present the prediction accuracies of the ALS and Alternating-IRLS algorithms with different noise addition mechanisms, and the interpretation of these results.

Datasets considered: Three different datasets were considered for the simulations:

1. A randomly generated synthetic dataset with different rank settings. The generation procedure is given in Subsection 5.2.
2. The MovieLens100k dataset (Harper and Konstan (2015)). The dataset consists of approximately 1000 users and 100000 ratings, accounting for a total of 5.1% (observable) entries.
3. The Sweet Recommender System dataset (Kidziński (2017)). This dataset contains 2000 users, 77 items, and over 45000 ratings which accounts for 39.19% of the total data. The visible entries had to be sub-sampled to match our desired observed fraction setting.

We measure the performance of the algorithms using the Root Mean Squared error metric which is calculated as $\left\|\mathbf{X} - \mathbf{U}\mathbf{V}^\top\right\|_F / (\sqrt{mn})$, where $X \in \mathbb{R}^{m \times n}$ [Chien *et al.* (2021), Liu *et al.* (2013)]. While experimenting with synthetic data, we vary the rank of $\mathbf{X}$, the noise variance, fraction of observed entries (also called the observable fraction or visible fraction) and evaluate the performance of the three noise addition mechanisms using both ALS and Alternating IRLS and decide the optimal course of action to collate and comparatively analyze results from the two real datasets.

Synthetic data:

$\mathbf{U} \in \mathbb{R}^{m \times r}$ and $\mathbf{V} \in \mathbb{R}^{n \times r}$ were randomly generated and scaled appropriately to get the desired entry range. They are then multiplied together to give the ground-truth low-rank matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$, where r is the rank of $\mathbf{M}$:

$$\begin{aligned} &\mathbf{U} \sim \mathcal{U}(0,\, 1)^{m \times r} \\ &\mathbf{V} \sim \mathcal{U}(0,\, 1)^{n \times r} \\ &\mathbf{M} = c \cdot \mathbf{U}\mathbf{V}^\top \end{aligned}$$

where c is a scaling constant to get the desired data range. $\mathbf{M}$ is then sub-sampled depending on the observed fraction setting to get the final incomplete data matrix. The hyperparameter settings used for the simulations are tabulated in Table 5.2.

Table 5.2: Simulation parameters

Parameter	Value
Number of alternating steps T	50
Number of IRLS iterations N	20
Regularization parameter λ	0.5
Trials for averaging	10
Ratings range	(1 - 5)

The RMSE results for the synthetic data of ranks 5, 10, and 20 are tabulated in Tables 5.3, 5.4, and 5.5 respectively. We also vary the type of noise addition mechanism, the variance of the added noise and visible fraction of the synthetic dataset to measure their impact on the performance of the two algorithms.

From the results in Tables 5.3, 5.4, and 5.5, we make two key observations:

1. The Huber mechanism gives more accurate results for a majority of the cases, and one speculation for this behaviour is that ALS and A-IRLS have the problem of converging to local minima. When the algorithm is stuck near a local minimum, the more heavily tailed Huber noise gives the algorithm a heavier momentum out of the local minimum than Gaussian noise, which is heavily centred around the mean. On the other hand, the Laplacian distribution has the longest tail of the three noise types and may provide excessive momentum to the algorithm when it is near the global minimum and cause it to converge to other local minima.
2. The A-IRLS procedure gives significant improvement over ALS for additive Huber noise in most cases, which is in line with theoretical expectations.

[1]No noise here indicates that the algorithm has no noise added for differential privacy. Inherent noise exists in all datasets.

Table 5.3: RMSE for synthetic dataset of rank **5**

Variance	Observed fraction	Algorithm	No noise [1]	Gaussian	Laplacian	Huber
1	5%	ALS	0.2040	0.3740	0.3701	**0.3569**
		A-IRLS	0.2039	0.3728	0.3939	**0.3640**
	10%	ALS	0.0948	0.2008	0.2017	**0.2002**
		A-IRLS	0.0950	0.2000	0.2013	**0.1995**
	15%	ALS	0.0567	0.1924	0.1926	**0.1923**
		A-IRLS	0.0567	0.1927	0.1924	**0.1901**
2	5%	ALS	0.2481	0.6388	0.6338	**0.6261**
		A-IRLS	0.2482	0.6370	0.6427	**0.6194**
	10%	ALS	0.0981	0.2203	0.2209	**0.2111**
		A-IRLS	0.0982	0.2194	0.2208	**0.2047**
	15%	ALS	0.0550	0.2018	0.2022	**0.1950**
		A-IRLS	0.0551	0.2016	0.2019	**0.1947**

Table 5.4: Results for Synthetic dataset of rank **10**

Variance	Observed fraction	Algorithm	No noise	Gaussian	Laplacian	Huber
1	5%	ALS	0.4058	0.6471	**0.6452**	0.6720
		A-IRLS	0.4059	0.6376	0.6888	**0.6479**
	10%	ALS	0.2317	**0.3015**	0.3029	0.3016
		A-IRLS	0.2317	**0.3023**	0.3037	0.3031
	15%	ALS	0.1602	0.2770	0.2774	**0.2768**
		A-IRLS	0.1596	0.2773	0.2771	**0.2767**
2	5%	ALS	0.3971	0.6454	0.6674	**0.6361**
		A-IRLS	0.3972	0.6835	0.7347	**0.6646**
	10%	ALS	0.2288	**0.3075**	0.3086	0.3098
		A-IRLS	0.2317	0.3086	0.3080	**0.3040**
	15%	ALS	0.1599	0.2836	0.2839	**0.2802**
		A-IRLS	0.1602	0.2822	0.2838	**0.2799**

Table 5.5: Results for Synthetic dataset of rank **20**

Variance	Observed fraction	Algorithm	No noise	Gaussian	Laplacian	Huber
1	5%	ALS	0.6902	1.5165	1.5330	**1.5142**
		A-IRLS	0.6886	1.5493	1.5715	**1.5191**
	10%	ALS	0.4622	0.5087	0.5113	**0.5083**
		A-IRLS	0.4600	**0.5054**	0.5189	0.5083
	15%	ALS	0.3961	0.4520	0.4529	**0.4516**
		A-IRLS	0.3957	0.4513	0.4526	**0.4494**
2	5%	ALS	0.6126	**1.8677**	1.8843	1.8770
		A-IRLS	0.6136	1.9546	1.9966	**1.9524**
	10%	ALS	0.4648	0.5222	0.5248	**0.5159**
		A-IRLS	0.4631	0.5220	0.5258	**0.5154**
	15%	ALS	0.3895	0.4470	0.4486	**0.4448**
		A-IRLS	0.3960	0.4474	0.4488	**0.4429**

Additionally, it can be observed that increasing the variance of the additive noise leads to a general decrease in performance, but this is heavily pronounced for data with lesser observed fraction. Additionally, this behaviour is more subtle for Huber noise than the Gaussian and Laplacian mechanisms, which suggests that the Huber mechanism is preferable for higher variance noise addition.

While analyzing the results for Huber noise, we can observe that A-IRLS gives better accuracy than ALS for higher variance values. One reason for this occurrence is that the Huber and Gaussian distributions have a similar shape for lower variance values (larger α values).

ALS gives a consistently better RMSE for Gaussian noise, which is in line with theoretical expectations. However, the Alternating-IRLS procedure gives a better performance than ALS for Laplacian noise in a majority of the cases. This makes A-IRLS more suitable for the Laplace mechanism.

Real datasets

After analyzing the results from the Synthetic data, we compare the results of ALS + Gaussian noise (ALS+G), A-IRLS + Laplacian noise (A-IRLS+L) and A-IRLS + Huber noise (A-IRLS+H) procedures for the MovieLens-100k and SweetRS datasets in Tables 5.6 and 5.7 respectively. Note that no noise is being added in the cases of plain ALS and A-IRLS and so they are not private algorithms.

Hyperparameter Settings for experiments with real data: We perform $T = 20$ iterations of ALS for Movielens (because of overfitting) and $T = 100$ for SweetRS. We set the model rank to 32 and regularization parameter $\lambda = 0.5$ for both the datasets. The A-IRLS procedure is run for $N = 10$ iterations. The results obtained are tabulated below:

Table 5.6: RMSE for MovieLens100k, Visible fraction = 5.1%

	ALS	**A-IRLS**	**ALS + G**	**A-IRLS + L**	**A-IRLS + H**
MSE	1.2463	1.2787	1.3883	1.3952	**1.3755**

Table 5.7: RMSE for SweetRS

Visible fraction	**ALS**	**A-IRLS**	**ALS + G**	**A-IRLS + L**	**A-IRLS + H**
5%	2.1333	2.1334	2.2103	2.2096	**2.2022**
10%	1.7103	1.7102	1.8830	1.8827	**1.8757**
15%	1.6881	1.6891	**1.7686**	1.7756	1.7694

From the results of the SweetRS dataset, we can observe the same trend in the performance of the A-IRLS procedure as observed in the synthetic data. We can also see that the differences in performance between private and non-private cases are orders of magnitude less as compared to Synthetic data. This may be because the Synthetic data was generated to adhere to a low-rank ground truth, which is just an assumption for real datasets.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we have explored the different noise addition mechanisms for Differential Privacy in the context of Low-Rank Matrix Completion. We introduced the novel Huber mechanism for noise addition and derived the privacy bounds for it in Chapter 4. We also introduced the new Alternating-IRLS optimization procedure for additive Huber noise in the same chapter which is tailored to give more accurate results for identical budget.

After interpreting the results from Chapter 5, we can conclude the following: 1) The Gaussian mechanism gives more accurate results, especially for data with a small fraction of observed entries, but the large privacy budget required works against its favour. 2) The Laplace mechanism gives a competent result for its low privacy budget requirement. But the heavy tail of the Laplacian distribution sometimes adds a large gradient leading to inconsistent model performance. 3) The Huber mechanism gives the best overall trade-off, with the dual advantages of a low privacy budget and high accuracy, particularly for stricter privacy budgets.

This thesis has only explored the merits of the Huber mechanism in the context of differentially private matrix completion. But we believe that it can be applied in a similar fashion to all privacy-preserving applications in general. Thus, it is necessary to analyze the benefits and drawbacks of the Huber mechanism in the context of other applications of differential privacy before reaching a generalized conclusion. Additionally, this thesis only uses the already established definitions of sensitivity, which may not be suitable for the Huber mechanism. It is worthwhile to explore the effects redefining sensitivity has on the various noise addition mechanisms.

REFERENCES

1. **Abadi, M.**, **A. Chu**, **I. Goodfellow**, **H. B. McMahan**, **I. Mironov**, **K. Talwar**, and **L. Zhang**, Deep learning with differential privacy. *In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016.

2. **Ahn, S.**, **A. Korattikara**, and **M. Welling** (2012). Bayesian posterior sampling via stochastic gradient fisher scoring. URL `https://arxiv.org/abs/1206.6380`.

3. **Bennett, J.** and **S. Lanning**, The netflix prize. 2007.

4. **Cai, J.-F.**, **E. J. Candes**, and **Z. Shen** (2008). A singular value thresholding algorithm for matrix completion. URL `https://arxiv.org/abs/0810.3286`.

5. **Calandrino, J.**, **A. Kilzer**, **A. Narayanan**, **E. Felten**, and **V. Shmatikov**, You might also like: Privacy risks of collaborative filtering. 2011.

6. **Candes, E. J.**, **Y. Eldar**, **T. Strohmer**, and **V. Voroninski** (2011). Phase retrieval via matrix completion. URL `https://arxiv.org/abs/1109.0573`.

7. **Candes, E. J.** and **Y. Plan** (2010). Matrix completion with noise. *Proceedings of the IEEE*, **98**(6), 925–936.

8. **Candes, E. J.** and **B. Recht** (2008). Exact matrix completion via convex optimization. URL `https://arxiv.org/abs/0805.4471`.

9. **Chien, S.**, **P. Jain**, **W. Krichene**, **S. Rendle**, **S. Song**, **A. Thakurta**, and **L. Zhang**, Private alternating least squares: Practical private matrix completion with tighter rates. *In International Conference on Machine Learning*. PMLR, 2021.

10. **Ding, B.**, **J. J. Kulkarni**, and **S. Yekhanin**, Collecting telemetry data privately. *In Advances in Neural Information Processing Systems 30*. 2017. URL `https://www.microsoft.com/en-us/research/publication/collecting-telemetry-data-privately/`.

11. **Dollinger, M. B.** and **R. G. Staudte** (1991). Influence functions of iteratively reweighted least squares estimators. *Journal of the American Statistical Association*, **86**(415), 709–716.

12. **Dwork, C.**, **F. McSherry**, **K. Nissim**, and **A. Smith**, Calibrating noise to sensitivity in private data analysis. volume Vol. 3876. 2006. ISBN 978-3-540-32731-8.

13. **Dwork, C.**, **F. McSherry**, and **K. Talwar**, The price of privacy and the limits of lp decoding. *In Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07. Association for Computing Machinery, New York, NY, USA, 2007. ISBN 9781595936318. URL `https://doi.org/10.1145/1250790.1250804`.

14. **Dwork, C.**, **A. Roth**, *et al.* (2014*a*). The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, **9**(3-4), 211–407.

15. **Dwork, C.**, **K. Talwar**, **A. Thakurta**, and **L. Zhang** (2014*b*). Analyze gauss: Optimal bounds for privacy-preserving principal component analysis. *Proceedings of the Annual ACM Symposium on Theory of Computing*, 11–20.

16. **Eckart, C.** and **G. M. Young** (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, **1**, 211–218.

17. **Erlingsson, U.**, **V. Pihur**, and **A. Korolova**, Rappor: Randomized aggregatable privacy-preserving ordinal response. CCS '14. Association for Computing Machinery, New York, NY, USA, 2014. ISBN 9781450329576. URL `https://doi.org/10.1145/2660267.2660348`.

18. **Frank, M.** and **P. Wolfe** (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, **3**(1-2), 95–110. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800030109`.

19. **Friedman, A.** and **A. Schuster**, Data mining with differential privacy. *In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010.

20. **Gowtham, R. A.**, **G. Muthukrishnan**, **T. Tholeti**, and **S. Kalyani** (2022). Introducing the huber mechanism for differentially private low-rank matrix completion. URL `https://arxiv.org/abs/2206.07910`.

21. **Hao, Y.**, **M. Cai**, and **L. Li** (2019). Drug repositioning via matrix completion with multi-view side information. *IET Systems Biology*, **13**.

22. **Harper, F. M.** and **J. A. Konstan** (2015). The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, **5**(4). ISSN 2160-6455. URL `https://doi.org/10.1145/2827872`.

23. **He, W.**, **H. Zhang**, **L. Zhang**, and **H. Shen** (2015). Total-variation-regularized low-rank matrix factorization for hyperspectral image restoration. *IEEE transactions on geoscience and remote sensing*, **54**(1), 178–188.

24. **Holland, P. W.** and **R. E. Welsch** (1977). Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, **6**(9), 813–827.

25. **Huber, P. J.** (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 73–101.

26. **Jain, P.**, **P. Netrapalli**, and **S. Sanghavi** (2012). Low-rank matrix completion using alternating minimization. *Proceedings of the Annual ACM Symposium on Theory of Computing*.

27. **Jain, P.**, **O. D. Thakkar**, and **A. Thakurta**, Differentially private matrix completion revisited. *In International Conference on Machine Learning*. PMLR, 2018.

28. **Kalyani, S.** and **K. Giridhar**, Mse analysis of the iteratively reweighted least squares algorithm when applied to m estimators. *In IEEE GLOBECOM 2007-IEEE Global Telecommunications Conference*. IEEE, 2007.

29. **Kapur, A.**, **K. Marwah**, and **G. Alterovitz** (2016). Gene expression prediction using low-rank matrix completion. *BMC Bioinformatics*, **17**.

30. **Kidziński,** (2017). Sweetrs: Dataset for a recommender systems of sweets. URL `https://arxiv.org/abs/1709.03496`.

31. **Koren, Y.** and **R. Bell**, *Advances in Collaborative Filtering*. 2015. ISBN 978-1-4899-7636-9, 77–118.

32. **Korolova, A.**, Privacy violations using microtargeted ads: A case study. 2010.

33. **Liu, C.**, **D. Wei**, **J. Xiang**, **F. Ren**, **L. Huang**, **J. Lang**, **G. Tian**, **Y. Li**, and **J. Yang** (2020). An improved anticancer drug-response prediction based on an ensemble method integrating matrix completion and ridge regression. *Molecular Therapy - Nucleic Acids*, **21**, 676–686. ISSN 2162-2531. URL `https://www.sciencedirect.com/science/article/pii/S216225312030192X`.

34. **Liu, Q.**, **F. Davoine**, **J. Yang**, **Y. Cui**, **Z. Jin**, and **F. Han** (2018). A fast and accurate matrix completion method based on qr decomposition and l2,1-norm minimization. *IEEE Transactions on Neural Networks and Learning Systems*, **PP**, 1–15.

35. **Liu, Y.**, **L. Jiao**, and **F. Shang** (2013). A fast tri-factorization method for low-rank matrix recovery and completion. *Pattern Recognition*, **46**, 163–173.

36. **Liu, Z.**, **Y.-X. Wang**, and **A. J. Smola** (2015). Fast differentially private matrix factorization. URL `https://arxiv.org/abs/1505.01419`.

37. **Lu, C.**, **J. Tang**, **S. Yan**, and **Z. Lin** (2015). Nonconvex nonsmooth low rank minimization via iteratively reweighted nuclear norm. *IEEE Transactions on Image Processing*, **25**(2), 829–839.

38. **Luo, X.**, **M. Zhou**, **Y. Xia**, and **Q. Zhu** (2014). An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics*, **10**(2), 1273–1284.

39. **Machanavajjhala, A.**, **D. Kifer**, **J. Abowd**, **J. Gehrke**, and **L. Vilhuber**, Privacy: Theory meets practice on the map. 2008.

40. **Recht, B.**, **W. Xu**, and **B. Hassibi**, Necessary and sufficient conditions for success of the nuclear norm heuristic for rank minimization. *In 2008 47th IEEE Conference on Decision and Control*. IEEE, 2008. URL `https://doi.org/10.1109%2Fcdc.2008.4739332`.

41. **Rendle, S.**, **L. Zhang**, and **Y. Koren** (2019). On the difficulty of evaluating baselines: A study on recommender systems. URL `https://arxiv.org/abs/1905.01395`.

42. **Rogers, R.**, **S. Subramaniam**, **S. Peng**, **D. Durfee**, **S. Lee**, **S. K. Kancha**, **S. Sahay**, and **P. Ahammad** (2020). Linkedin's audience engagements api: A privacy preserving data analytics system at scale. URL `https://arxiv.org/abs/2002.05839`.

43. **Song, S.**, **K. Chaudhuri**, and **A. D. Sarwate**, Stochastic gradient descent with differentially private updates. *In 2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013.

44. **Welling, M.** and **Y. W. Teh**, Bayesian learning via stochastic gradient langevin dynamics. *In Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11. Omnipress, Madison, WI, USA, 2011. ISBN 9781450306195.

45. **Xia, C.**, **J. Hua**, **W. Tong**, and **S. Zhong** (2020). Distributed k-means clustering guaranteeing local differential privacy. *Computers & Security*, **90**, 101699.

LIST OF PAPERS BASED ON THESIS

1. **Gowtham, R. A.**, **G. Muthukrishnan**, **T. Tholeti**, and **S. Kalyani** (2022). Introducing the huber mechanism for differentially private low-rank matrix completion. URL `https://arxiv.org/abs/2206.07910`

APPENDIX A

Derivation of privacy bounds for the Huber mechanism

For getting the privacy bounds of Huber noise, we need to derive the bounds case-wise and then combine them to obtain the universal bound.

A.1 Case 1: $\Delta f \leq 2\alpha$

We go over all the intervals of x in order and find the upper bound of the function $g(x)$.

i. For $x < -\Delta f - \alpha$, both $|x|$ and $|x + \Delta f|$ are greater than α, both x and $x + \Delta f$ are negative :

$$g_1(x) = \alpha \cdot \left(|x + \Delta f| - \frac{\alpha}{2}\right) - \alpha \cdot \left(|x| - \frac{\alpha}{2}\right) \tag{A.1}$$

$$= \alpha \cdot \left(-x - \Delta f - \frac{\alpha}{2}\right) - \alpha \cdot \left(-x - \frac{\alpha}{2}\right) \tag{A.2}$$

$$= -\alpha \cdot \Delta f \tag{A.3}$$

Since the value of $g_1(x)$ is a constant, the maximum of $g_1(x)$ in this range ($g_{1_{max}}$) is $-\alpha \cdot \Delta f$ itself.

ii. For $-\Delta f - \alpha \leq x \leq -\alpha$, $|x|$ remains greater than α whereas $|x + \Delta f|$ becomes less than α :

$$g_2(x) = \frac{(x + \Delta f)^2}{2} - \alpha \cdot \left(|x| - \frac{\alpha}{2}\right) \tag{A.4}$$

$$= \frac{x^2 + 2x\Delta f + \Delta f^2}{2} - \alpha \cdot \left(-x - \frac{\alpha}{2}\right) \tag{A.5}$$

$$= \frac{x^2 + 2x(\alpha + \Delta f) + \Delta f^2 + \alpha^2 + 2\alpha\Delta f}{2} - \alpha \cdot \Delta f \tag{A.6}$$

$$= \frac{(x + \alpha + \Delta f)^2}{2} - \alpha \cdot \Delta f \tag{A.7}$$

The minimum value of $g_2(x)$ occurs at $x = -\alpha - \Delta f$. The function is then monotonically increasing as we approach $-\alpha$, i.e. the maximum in this range (say $g_{2_{max}}$ occurs at $x = -\alpha$, which is:

$$g_{2_{max}} = g_2(x)\big|_{x=-\alpha} \tag{A.8}$$

$$= \frac{(-\alpha + \alpha + \Delta f)^2}{2} - \alpha \cdot \Delta f \tag{A.9}$$

$$= \frac{\Delta f \cdot (\Delta f - 2\alpha)}{2} \tag{A.10}$$

$$\leq 0 \tag{A.11}$$

Since $\Delta f \leq 2\alpha$ by the case definition.

iii. For $-\alpha < x \leq \alpha - \Delta f$, $|x|$ becomes less than α and $|x + \Delta f|$ remains less than α, hence :

$$g_3(x) = x\Delta f + \frac{\Delta f^2}{2} \tag{A.12}$$

The maximum value in this range occurs at $x = \alpha - \Delta f$ as $g_3(x)$ is monotonically increasing (as the slope $\Delta f \geq 0$ by definition). Therefore,

$$g_{3_{max}} = g_3(x)\big|_{x=\alpha-\Delta f} \tag{A.13}$$

$$= \alpha \cdot \Delta f - \frac{\Delta f^2}{2} \tag{A.14}$$

$$\leq \alpha \cdot \Delta f \tag{A.15}$$

Since $\Delta f \geq 0$ by definition. In this case, $g_{3_{max}}$ is non-negative and upper bounded by $\alpha \cdot \Delta f$.

iv. For $\alpha - \Delta f < x \leq \alpha$, $|x|$ remains less than α whereas $|x + \Delta f|$ becomes greater than α. Now, $x + \Delta f$ also becomes positive, so :

$$g_4(x) = \alpha \cdot \left(|x + \Delta f| - \frac{\alpha}{2}\right) - \frac{x^2}{2} \tag{A.16}$$

$$= \alpha \cdot \left(x + \Delta f - \frac{\alpha}{2}\right) - \frac{x^2}{2} \tag{A.17}$$

$$= \alpha \cdot \Delta f - \frac{1}{2}(x - \alpha)^2 \tag{A.18}$$

The maximum value in this range occurs at $x = \alpha$ as $(x - \alpha)^2$ has a global minimum at $x = \alpha$. Hence, $g_{4_{max}}$ for this range is :

$$g_{4_{max}} = \alpha \cdot \Delta f \tag{A.19}$$

v. For $x > \alpha$, both $|x|$ and $|x + \Delta f|$ are greater than α, both x and $x + \Delta f$ are positive :

$$g_5(x) = \alpha \cdot \left(|x+\Delta f| - \frac{\alpha}{2}\right) - \alpha \cdot \left(|x| - \frac{\alpha}{2}\right) \tag{A.20}$$

$$= \alpha \cdot \left(x+\Delta f - \frac{\alpha}{2}\right) - \alpha \cdot \left(x - \frac{\alpha}{2}\right) \tag{A.21}$$

$$= \alpha \cdot \Delta f \tag{A.22}$$

Since the value of $g_5(x)$ comes out to be a constant, the maximum of $g_5(x)$ in this range ($g_{5_{max}}$) is $\alpha \cdot \Delta f$ itself.

Piecing the values together, we get the piece-wise defined function $\exp(H_\alpha(x+\Delta f) - H_\alpha(x))$ as :

$$\exp(g(x)) = \begin{cases} e^{-\alpha\cdot\Delta f} & ;\ x < -\Delta f - \alpha, \\ e^{\frac{(x+\alpha+\Delta f)^2}{2} - \alpha\cdot\Delta f} & ;\ -\Delta f - \alpha \le x \le -\alpha, \\ e^{x\cdot\Delta f + \frac{\Delta f^2}{2}} & ;\ -\alpha < x \le \alpha - \Delta f, \\ e^{\alpha\cdot\Delta f - \frac{1}{2}(x-\alpha)^2} & ;\ \alpha - \Delta f < x \le \alpha, \\ e^{\alpha\cdot\Delta f} & ;\ x > \alpha. \end{cases} \tag{A.23}$$

The overall upper bound g_{max} for Case 1 can be computed as

$$\implies g_{max} = \max(g_{1_{max}}, g_{2_{max}}, g_{3_{max}}, g_{4_{max}}, g_{5_{max}}) \tag{A.24}$$

$$\implies g_{max} = \alpha \cdot \Delta f \tag{A.25}$$

So, the upper bound of $\exp(H_\alpha(x+\Delta f) - H_\alpha(x))$ for case 1, say H_{max1} is

$$\implies H_{max1} = \exp(g_{max})$$

$$\implies H_{max1} = \exp(\alpha \cdot \Delta f)$$

A.2 Case 2: $\Delta f > 2\alpha$

Similar to case 1, we compute the upper bounds of $g(x)$ for different intervals of x.

i. For $x < -\Delta f - \alpha$, $g_1(x)$ and $g_{1_{max}}$ are identical to those in Case 1.

ii. For $-\Delta f - \alpha \le x \le \alpha - \Delta f$, $|x|$ remains greater than α whereas $|x+\Delta f|$

becomes less than α :

$$g_2(x) = \frac{(x+\Delta f)^2}{2} - \alpha \cdot \left(|x| - \frac{\alpha}{2}\right) \tag{A.26}$$

$$= \frac{x^2 + 2x\Delta f + \Delta f^2}{2} - \alpha \cdot \left(-x - \frac{\alpha}{2}\right) \tag{A.27}$$

$$= \frac{(x+\alpha+\Delta f)^2}{2} - \alpha \cdot \Delta f \tag{A.28}$$

The maximum value of $g_2(x)$ occurs at $x = \alpha - \Delta f$ as the minimum is at $x = -\alpha - \Delta f$ and the function is then monotonically increasing as we approach $x = \alpha - \Delta f$. Hence,

$$g_{2_{max}} = g_2(x)\big|_{x=\alpha-\Delta f} \tag{A.29}$$

$$= \frac{(\alpha - \Delta f + \alpha + \Delta f)^2}{2} - \alpha \cdot \Delta f \tag{A.30}$$

$$= \alpha \cdot (2\alpha - \Delta f) \tag{A.31}$$

$$\leq 0 \tag{A.32}$$

since $\Delta f > 2\alpha$ by the case definition.

iii. For $\alpha - \Delta f < x \leq -\alpha$, $|x|$ remains greater than α and $|x + \Delta f|$ becomes greater than α. Also, note that x is negative but $x + \Delta f$ becomes non-negative, hence :

$$g_3(x) = \alpha \cdot \left(|x+\Delta f| - \frac{\alpha}{2}\right) - \alpha \cdot \left(|x| - \frac{\alpha}{2}\right) \tag{A.33}$$

$$= \alpha \cdot \left(x + \Delta f - \frac{\alpha}{2}\right) - \alpha \cdot \left(-x - \frac{\alpha}{2}\right) \tag{A.34}$$

$$= \alpha \cdot (2x + \Delta f) \tag{A.35}$$

The maximum value in this range occurs at $x = -\alpha$ as $g_3(x)$ is monotonically increasing (as the slope $2\alpha \geq 0$ by definition of α). Therefore,

$$g_{3_{max}} = g_3(x)\big|_{x=-\alpha} \tag{A.36}$$

$$= \alpha \cdot \Delta f - 2\alpha^2 \tag{A.37}$$

$$\leq \alpha \cdot \Delta f \tag{A.38}$$

since $2\alpha^2 \geq 0$. In this case too, $g_{3_{max}}$ is non-negative and upper bounded by $\alpha \cdot \Delta f$.

iv. For $-\alpha < x \leq \alpha$, $|x|$ becomes less than α. So :

$$g_4(x) = \alpha \cdot \left(|x+\Delta f| - \frac{\alpha}{2}\right) - \frac{x^2}{2} \tag{A.39}$$

$$= \alpha \cdot \left(x + \Delta f - \frac{\alpha}{2}\right) - \frac{x^2}{2} \tag{A.40}$$

$$= \alpha \cdot \Delta f - \frac{1}{2}(x - \alpha)^2 \tag{A.41}$$

The maximum value in this range occurs at $x = \alpha$ as before. Therefore, the maximum $g_{4_{max}}$ for this range is :

$$g_{4_{max}} = \alpha \cdot \Delta f \tag{A.42}$$

v. For $x > \alpha$ too, $g_5(x)$ and $g_{5_{max}}$ are identical to those in Case 1.

Here, the piece-wise defined function $\exp(H_\alpha(x + \Delta f) - H_\alpha(x))$ as

$$\exp(g(x)) = \begin{cases} e^{-\alpha \cdot \Delta f} & ; x < -\Delta f - \alpha, \\ e^{\frac{(x+\alpha+\Delta f)^2}{2} - \alpha \cdot \Delta f} & ; -\Delta f - \alpha \leq x \leq \alpha - \Delta f, \\ e^{\alpha \cdot (2x + \Delta f)} & ; \alpha - \Delta f < x \leq -\alpha, \\ e^{\alpha \cdot \Delta f - \frac{1}{2}(x-\alpha)^2} & ; -\alpha < x \leq \alpha, \\ e^{\alpha \cdot \Delta f} & ; x > \alpha. \end{cases} \tag{A.43}$$

The overall upper bound g_{max} for Case 2 is

$$\implies g_{max} = \max(g_{1_{max}}, g_{2_{max}}, g_{3_{max}}, g_{4_{max}}, g_{5_{max}}) \tag{A.44}$$

$$\implies g_{max} = \alpha \cdot \Delta f \tag{A.45}$$

So, the upper bound of $\exp(H_\alpha(x + \Delta f) - H_\alpha(x))$ for case 2, say H_{max2} is also:

$$\implies H_{max2} = \exp(g_{max})$$

$$\implies H_{max2} = \exp(\alpha \cdot \Delta f)$$

Hence, the overall upper bound H_{max} is given as:

$$H_{max} = \max(H_{max1}, H_{max2}) = \exp(\alpha \cdot \Delta f) \tag{A.46}$$

And, the upper bound of $\mathcal{G}_\alpha(x)$, as evidenced by the above proof is $(\alpha \cdot \Delta f)$.

APPENDIX B

Deriving the CDF and Inverse-CDF of the Huber Distribution

B.1 The Huber CDF

The CDF of any probability distribution $p_X(x)$ is defined as:

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} p_X(t) \cdot dt \tag{B.1}$$

Since the Huber PDF is defined in parts, it is necessary to split the integral into cases and evaluate them separately:

CASE 1: $x \in (-\infty, -\alpha)$

$$F_X(x) = \int_{-\infty}^{x} \frac{1}{N(\alpha)} \cdot e^{-\alpha(-t-\alpha/2)}\, dt \tag{B.2}$$

$$= \frac{1}{N(\alpha)} \cdot \int_{-\infty}^{x} e^{\alpha(t+\alpha/2)}\, dt \tag{B.3}$$

Setting $e^{\alpha\cdot(t+\alpha/2)} = p \implies \alpha \cdot p = dp$. Substituting this in Equation B.3, we get:

$$F_X(x) = \frac{1}{\alpha \cdot N(\alpha)} \cdot \int dp \tag{B.4}$$

$$= \frac{1}{\alpha \cdot N(\alpha)} \cdot e^{\alpha(t+\alpha/2)}|_{-\infty}^{x} \tag{B.5}$$

$$\implies F_X(x) = \frac{1}{\alpha \cdot N(\alpha)} \cdot \left(e^{\alpha(x+\alpha/2)}\right) \tag{B.6}$$

CASE 2: $x \in [-\alpha,\, \alpha\,]$

$$F_X(x) = \frac{1}{N(\alpha)} \left(\int_{-\infty}^{-\alpha} e^{-\alpha(-t-\alpha/2)} \cdot dt + \int_{-\alpha}^{x} e^{-t^2/2} \cdot dt \right) \tag{B.7}$$

$$= \frac{1}{\alpha \cdot N(\alpha)} \cdot e^{-\alpha^2/2} \; + \frac{1}{N(\alpha)} \cdot \int_{-\alpha}^{x} e^{-t^2/2} \cdot dt \tag{B.8}$$

$$\implies F_X(x) = \frac{e^{-\alpha^2/2}}{\alpha \cdot N(\alpha)} \; + \frac{1}{N(\alpha)} \cdot \sqrt{\frac{\pi}{2}} \cdot \left(\text{erf}\left(\frac{x}{\sqrt{2}}\right) - \text{erf}\left(\frac{-\alpha}{\sqrt{2}}\right) \right) \tag{B.9}$$

CASE 3: $x \in (\alpha,\, \infty)$

$$F_X(x) = \frac{1}{N(\alpha)} \cdot \left(\int_{-\infty}^{-\alpha} e^{\alpha(t+\alpha/2)} \cdot dt + \int_{-\alpha}^{\alpha} e^{-t^2/2} \cdot dt + \int_{\alpha}^{x} e^{-\alpha(t-\alpha/2)} \cdot dt \right) \tag{B.10}$$

$$= \frac{1}{\alpha \cdot N(\alpha)} \cdot \left(e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\alpha/\sqrt{2}\right) + e^{-\alpha^2/2} - e^{-\alpha(x-\alpha/2)} \right) \tag{B.11}$$

$$\implies F_X(x) = \frac{1}{\alpha \cdot N(\alpha)} \cdot \left(2e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\alpha/\sqrt{2}\right) - e^{-\alpha(x-\alpha/2)} \right) \tag{B.12}$$

Combining the results of all three Cases, the Huber noise CDF is given as:

$$F_X(x) = \begin{cases} \frac{1}{\alpha \cdot N(\alpha)} \cdot \left(e^{\alpha(x+\alpha/2)} \right), & ;\, x \leq -\alpha \\ \frac{1}{\alpha \cdot N(\alpha)} \cdot e^{-\alpha^2/2} \; + \frac{1}{N(\alpha)} \cdot \sqrt{\frac{\pi}{2}} \cdot \left(\text{erf}\left(\frac{x}{\sqrt{2}}\right) - \text{erf}\left(\frac{-\alpha}{\sqrt{2}}\right) \right), & ;\, -\alpha \leq x \leq \alpha \\ \frac{1}{\alpha \cdot N(\alpha)} \cdot \left(2e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\alpha/\sqrt{2}\right) - e^{-\alpha(x-\alpha/2)} \right), & ;\, x > \alpha \end{cases} \tag{B.13}$$

B.2 The Inverse of the Huber-CDF

Since the Huber CDF is also defined in parts (Equation B.13), we need to invert it on a case-by-case basis too.

CASE 1: $x \in (-\infty, -\alpha)$

To begin the inverting process, we set *y* to:

$$y = \frac{1}{\alpha \cdot N(\alpha)} \cdot \left(e^{\alpha(x+\alpha/2)}\right) \tag{B.14}$$

$$\implies \frac{1}{\alpha} \cdot \log\left(\alpha \cdot N(\alpha) \cdot y\right) = \alpha \cdot (x + \alpha/2) \tag{B.15}$$

$$\implies x = \frac{1}{\alpha} \cdot \log\left(\alpha \cdot N(\alpha) \cdot y\right) - \frac{\alpha}{2} \tag{B.16}$$

$\forall\, y \in \left(0,\, \frac{1}{\alpha \cdot N(\alpha)} \cdot e^{-\alpha^2/2}\right)$.

CASE 2: $x \in [-\alpha,\, \alpha]$

Like before, we set y to:

$$y = \frac{1}{\alpha \cdot N(\alpha)} \cdot e^{-\alpha^2/2} \; + \frac{1}{N(\alpha)} \cdot \sqrt{\frac{\pi}{2}} \cdot \left(\text{erf}\left(\frac{x}{\sqrt{2}}\right) - \text{erf}\left(\frac{-\alpha}{\sqrt{2}}\right)\right) \tag{B.17}$$

$$\left(y - \frac{1}{\alpha \cdot N(\alpha)} \cdot e^{-\alpha^2/2}\right) \cdot N(\alpha) \cdot \sqrt{\frac{2}{\pi}} + \text{erf}\left(\frac{-\alpha}{\sqrt{2}}\right) = \text{erf}\left(\frac{x}{\sqrt{2}}\right) \tag{B.18}$$

$$\implies x = \sqrt{2} \cdot \text{inv\_erf}\left(\left(y \cdot N(\alpha) - \frac{1}{\alpha} \cdot e^{-\alpha^2/2}\right) \cdot \sqrt{\frac{2}{\pi}} + \text{erf}\left(\frac{-\alpha}{\sqrt{2}}\right)\right) \tag{B.19}$$

$\forall\, y \in \left[\frac{1}{\alpha \cdot N(\alpha)} \cdot e^{-\alpha^2/2},\, \frac{1}{\alpha \cdot N(\alpha)} \cdot \left(e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right)\right)\right]$. Here, inv_erf(.) refers to the inverse-error function.

CASE 3: $x \in (\alpha,\, \infty)$

Again, we set y to:

$$y = \frac{1}{\alpha \cdot N(\alpha)} \cdot \left(2e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\alpha/\sqrt{2}\right) - e^{-\alpha(x-\alpha/2)}\right) \tag{B.20}$$

$$\log\left(-y \cdot \alpha \cdot N(\alpha) + 2e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right)\right) = \alpha \cdot \left(x - \frac{\alpha}{2}\right) \tag{B.21}$$

$$\implies x = \frac{1}{\alpha} \cdot \log\left(2e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right) - y \cdot \alpha \cdot N(\alpha)\right) + \frac{\alpha}{2} \tag{B.22}$$

$\forall\, y \in \left(\frac{1}{\alpha \cdot N(\alpha)} \cdot \left(e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right)\right),\, 1\right)$.

Compiling the resulting from all three cases, the inverse CDF of Huber noise is defined as follows. First, we define y_1 and y_2 as:

$$y_1 = \frac{1}{\alpha \cdot N(\alpha)} \cdot e^{-\alpha^2/2} \tag{B.23}$$

$$y_2 = \frac{1}{\alpha \cdot N(\alpha)} \cdot \left(e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right)\right) \tag{B.24}$$

The Inverse CDF for Mixed-Huber Noise is:

$$F_Y^{-1}(y) = \begin{cases} \frac{1}{\alpha} \cdot \log\left(\alpha \cdot N(\alpha) \cdot y\right) - \frac{\alpha}{2} & ;\, y \in (0,\, y_1) \\ \sqrt{2} \cdot \text{inv\_erf}\left(\left(y \cdot N(\alpha) - \frac{1}{\alpha} \cdot e^{-\alpha^2/2}\right) \cdot \sqrt{\frac{2}{\pi}} + \text{erf}\left(\frac{-\alpha}{\sqrt{2}}\right)\right) & ;\, y \in [\,y_1,\, y_2\,] \\ \frac{1}{\alpha} \cdot \log\left(2e^{-\alpha^2/2} + \alpha\sqrt{2\pi} \cdot \text{erf}\left(\frac{\alpha}{\sqrt{2}}\right) - y \cdot \alpha \cdot N(\alpha)\right) + \frac{\alpha}{2} & ;\, y \in (y_2,\, 1) \end{cases} \tag{B.25}$$

where inv_erf(.) is the inverse error function, given as:

$$\text{inv\_erf}(x) = \sum_{k=0}^{\infty} \frac{c_k}{2k+1} \left(\frac{\sqrt{\pi}}{2} x\right)^{2k+1} \tag{B.26}$$

where $c_0 = 1$ and c_k is calculated using the recursive procedure given below:

$$c_k = \sum_{m=0}^{k-1} \frac{c_m c_{k-1-m}}{(m+1)(2m+1)} \tag{B.27}$$