공학박사학위논문

Privacy-preserving Machine Learning for Protecting Sensitive Information

민감한 정보를 보호할 수 있는 프라이버시 보존 기계학습 기술 개발

2022 년 8 월

서울대학교 대학원

산업공학과

변 준 영

Privacy-preserving Machine Learning for Protecting Sensitive Information

민감한 정보를 보호할 수 있는 프라이버시 보존 기계학습 기술 개발

지도교수 이 재 욱

이 논문을 공학박사 학위논문으로 제출함

2022 년 6 월

서울대학교 대학원

산업공학과

변 준 영

변준영의 공학박사 학위논문을 인준함

2022 년 8 월

위 원 장	조 성 준	(인)
부위원장	이 재 욱	(인)
위 원	박 우 진	(인)
위 원	박 새 롬	(인)
위 원	이 우 진	(인)

Abstract

Privacy-preserving Machine Learning for Protecting Sensitive Information

Junyoung Byun
Department of Industrial Engineering
The Graduate School
Seoul National University

Recent development of artificial intelligence systems has been driven by various factors such as the development of new algorithms and the the explosive increase in the amount of available data. In the real-world scenarios, individuals or corporations benefit by providing data for training a machine learning model or the trained model. However, it has been revealed that sharing of data or the model can lead to invasion of personal privacy by leaking personal sensitive information.

In this dissertation, we focus on developing privacy-preserving machine learning methods which can protect sensitive information. Homomorphic encryption can protect the privacy of data and the models because machine learning algorithms can be applied to encrypted data, but requires much larger computation time than conventional operations. For efficient computation, we take two approaches. The first is to reduce the amount of computation in the training phase. We present an efficient training algorithm by encrypting only few important information. In specific, we

develop a ridge regression algorithm that greatly reduces the amount of computation when one or two sensitive variables are encrypted. Furthermore, we extend the method to apply it to classification problems by developing a new logistic regression algorithm that can maximally exclude searching of hyper-parameters that are not suitable for machine learning with homomorphic encryption. Another approach is to apply homomorphic encryption only when the trained model is used for inference, which prevents direct exposure of the test data and the model information. We propose a homomorphic-encryption-friendly algorithm for inference of support based clustering.

Though homomorphic encryption can prevent various threats to data and the model information, it cannot defend against secondary attacks through inference APIs. It has been reported that an adversary can extract information about the training data only with his or her input and the corresponding output of the model. For instance, the adversary can determine whether specific data is included in the training data or not. Differential privacy is a mathematical concept which guarantees defense against those attacks by reducing the impact of specific data samples on the trained model. Differential privacy has the advantage of being able to quantitatively express the degree of privacy, but it reduces the utility of the model by adding randomness to the algorithm. Therefore, we propose a novel method which can improve the utility while maintaining the privacy of differentially private clustering algorithms by utilizing Morse theory.

The privacy-preserving machine learning methods proposed in this paper can complement each other to prevent different levels of attacks. We expect that our methods can construct an integrated system and be applied to various domains

where machine learning involves sensitive personal information.

Keywords: Privacy-preserving machine learning, Homomorphic encryption, Differential priavcy

Student Number: 2017-28535

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation of the Dissertation

In recent years, research on machine learning has shown an unprecedented growth both quantitatively and qualitatively. It has been widely recognized that machine learning can be a key technology in various fields including computer vision, natural language processing, marketing and finance, and can even achieve a better performance than human can in several applications. Machine learning is now so popular that it is harder to find fields where it has never been applied. The main reasons behind the success of machine learning are the development of computational processing power, the development of various algorithms, and the increase in the amount of data accumulated through internet.

In the real world, machine learning applications usually include the transfer of data or model information from one subject to another. Essentially, this is because it is not common to have sufficient amounts of data and computational power to train machine learning models at the same time. When data and computational power are separated, data should be delivered to the computing source which conducts the training. In cases where the ownership of the trained model is with the data owner, the model should be passed back to the data owner. Also, the trained model can be

used for inference on external data to create further value, and again, the data or model must be passed on to others.

In this sense, it has been pointed out that the leakage of sensitive information may occur in the process of machine learning [3, 81, 125, 176, 177]. Because the value of the data and the trained model is very high, attempts to steal their information can occur at any time. The most intuitive threat is the direct exposure of raw data or model, which naturally leads to infringement of intellectual property rights. In 2021, Facebook, one of the world's most reputable artificial intelligence companies, admitted that about 533 millions of their user profiles had been compromised. The leaked information was disclosed on a hacking site and anyone could access it for free. As in this case, exposure of raw data is even more detrimental because the data is likely to contain sensitive personal information that can characterize an individual. Therefore, when any information is transmitted to another entity, it must be processed so that the information is not directly revealed.

On the other hand, there are cases where information is indirectly leaked from the trained machine learning model. In 2020, a Korean company Scatterlab launched an artificial intelligence-based conversation service called LUDA. However, it was soon found out that personal information such as a person's name or address, which was contained in the actual conversations used for training LUDA, could be leaked through specific questions. Afterwards, Scatterlab stopped the service and promised a thorough de-identification process. However, due to the nature of text data which is unstructured, complete de-identification is considered to be very difficult.

Based on these cases, the various privacy threats that occur in the machine learning process can be divided in to several categories. The first is direct access

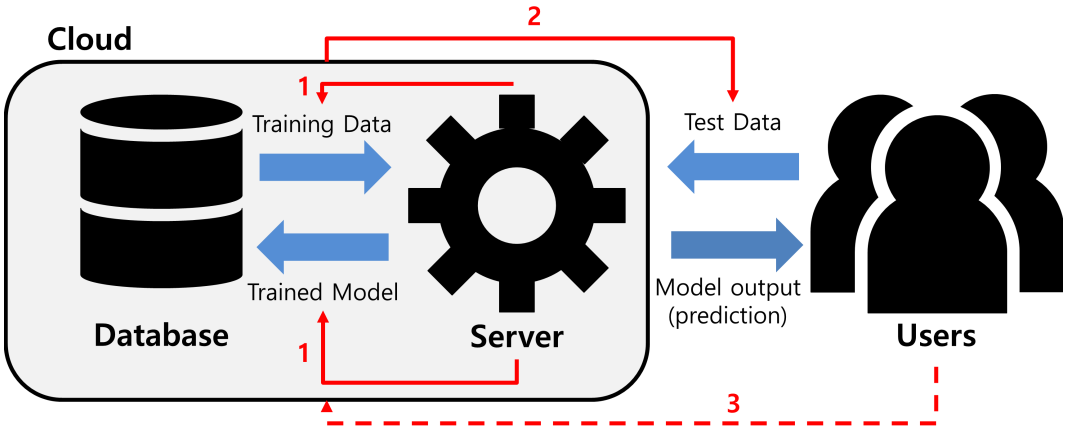


Figure 1.1: Different levels of privacy threats can happen in either the training phase or the inference phase of machine learning. Bold arrows indicate direct threats, and dotted arrows indicate indirect threats.

to data or model information. In the case of Facebook, the attack was caused by external hacking, but it can be pointed out that the problem is that the company retains a massive personal information as raw data. In other words, the transmission of data or model information to others may itself be the biggest threat. As mentioned above, such transfer of information can occur either in the training phase or in the inference phase. We refer to the direct information exposure that occurs in the training phase and the inference phase as level 1 threat and level 2 threat, respectively. Meanwhile, as seen in the case of LUDA, information leakage may occur in the process of using the trained model by new users. We refer to this indirect information exposure as level 3 threat. Figure 1.1 briefly presents different threats.

Among various data processing techniques which have been proposed to deal with the threats on sensitive information, cryptographic protocols can be expected to provide the highest level of security by encrypting the sensitive information.

Homomorphic encryption is a powerful cryptographic tool which enables basic computations on encrypted data [75], which can be extended to more complex functions including machine learning models. When applying homomorphic encryption to machine learning, the data owner encrypts his or her data and transfers the encrypted data to the computing source. The computing source then obtains the model by conducting training on the encrypted data. In this way, not only does the information of the data not leak to the computing source, but the trained model is obtained encrypted so that no unauthorized one can steal the information of the model. Therefore, homomorphic encryption can defend against level 1 threat.

Despite the desirable properties of homomorphic encryption, its computation time makes it difficult to port homomorphic encryption to existing machine learning algorithms. Therefore, studies on the training phase have been mainly conducted on simple models such as logistic regression [7, 9, 29, 101] and linear regression [64, 144]. [140] proposed an homomorphic-encryption-friendly algorithm for training support vector machines, but it is assumed that the kernel matrix can be constructed before encryption, which can reduce the level of security to a certain extent. In the studies on logistic regression and support vector machines, training is done through gradient descent. However, gradient descent methods involve various hyperparameters, including learning rate and number of epochs. If not encrypted, these parameters are determined through validation. However, validation is not possible when data is encrypted, because the intermediate product must be decrypted for validation, which means information leakage. Though [103] proposed an approximate Newton's method to remove learning rate, still other parameters should be validated, and the convergence of the approximate method was not thoroughly investigated.

Studies on linear regression may or may not require gradient descent, but for the latter on-line communication is required between participants to securely calculate the inverse matrix.

For rather complex models such as neural networks [66, 17, 20, 120] and kernel support vector machines [83, 145, 11], because training takes too much time, their inference phases have been mainly studied. In machine-learning-as-a-service models which have been common in various applications such as recommendation and healthcare services, clients send their data to the cloud where inferences are made on the data through pre-trained machine learning models. By encrypting the clients' data, their sensitive information is not exposed to the cloud. Also, because the model parameters need not to be encrypted, the inference can be made more efficiently. In this case, level 2 privacy can be preserved.

Homomorphic encryption can guarantee the privacy of the sensitive information against direct exposure, but it cannot defend against every kind of attack. Especially, homomorphic encryption cannot prevent indirect leakage of information from the model output, because the client should be able to decrypt the model output even if homomorphic encryption is used for secure computation. In other words, homomorphic encryption has weakness that it cannot defend against level 3 threats. There have been studied various attacks using model output, where a malicious client attempts to acquire some information about the model or the training data with his or her input and the corresponding model output. Those attacks can be classified according to the type of information the adversary wants to obtain, including model extraction attacks [167, 132, 136], model inversion attacks [59, 185], and membership inference attacks [156, 149].

Differential privacy provides a formal privacy guarantee, which can defend against the attacks through model output. By its definition, differential privacy limits the change in the model due to the change in the training data. Therefore, when the trained model satisfies differential privacy, it becomes naturally resistant to level 3 threat. The advantages of differential privacy is that the level of privacy can be quantitatively determined, and it is always possible to satisfy the desired degree of privacy. Recently, there have been massive studies on differentially private machine learning, ranging from empirical risk minimization [27] to deep learning [1, 137]. In order to satisfy differential privacy, randomness must be added to the algorithm, and the higher the desired level of privacy, the larger the amount of randomness is required, which harms the performance of the model. Therefore, research on differentially private machine learning aims to maximize the level of privacy while maintaining the performance of the model.

To summarize, various kinds of threats on sensitive information can occur in the training or inference phase of machine learning, and a single defense method cannot protect against all of them. Therefore, in this dissertation we propose methods each of which can defend against each kind of attack. To prevent direct exposure of data and model, we propose novel and homomorphic-encryption-friendly methods for machine learning training and inference, respectively. Then we present a novel differentially private method which can hide the information of the sensitive information from the model output.

1.2 Aims of the Dissertation

This dissertation aims to develop privacy-preserving machine learning methods which can defend against various attacks on sensitive information. We first propose an efficient training of ridge regression method with homomorphic encryption, which protects level 1 privacy. Extending the ridge regression method, we then propose a logistic regression method which does not require searching for various hyperparameters. We then propose a homomorphic-encryption-friendly inference of support based clustering method, which protects level 2 privacy. Finally, to protect level 3 privacy, we present a novel differentially clustering method that enhances utility of the existing methods while preserving their privacy. The summarization of the research is as follows:

1. **Efficient Homomorphic Encryption Framework for Ridge Regression (Chapter 3)** To overcome the limitations on existing studies, we propose an efficient linear regression training method which does not require gradient descent. To achieve our goal, we exploit the fact that not all information in the data is critical enough to require encryption. By partially encrypting sensitive information, our method can examine the exact solution of ridge regression without gradient descent. Using Sherman-Morrison-Woodbury formula for matrix inversion, the operations on the ciphertext and the operations on the plaintext can be further separated in the process of calculating the inverse matrix. When encrypting only sensitive information, there is another privacy concern that an adversary can infer the sensitive information with unencrypted, non-sensitive information. Inspired by generative adversar-

ial perturbation [143] originally proposed for generating adversarial examples, we present an effective method to prevent those attacks.

2. **Parameter-free Homomorphic-encryption-friendly Logistic Regression (Chapter 4)** Current approaches on the training of encrypted machine learning have relied heavily on hyper-parameter selection, which should be avoided owing to the extreme difficulty of conducting validation on encrypted data. To overcome the limitation, we propose an effective privacy-preserving logistic regression method that is free from the approximation of the sigmoid function and other hyper-parameter selection. By training a teacher model with unencrypted data and inferring encrypted data with the teacher model, a logistic regression model is transformed into the corresponding ridge regression for the logit function. Then we train an encrypted ridge regression to predict the logit as demonstrated above. In addition, we propose a homomorphic-encryption-friendly mean-matching method which aims to reduce the difference in the distributions of encrypted and unencrypted data.

3. **Homomorphic-encryption-friendly Evaluation for Support Vector Clustering (Chapter 5)** Although efficient algorithms have been proposed in the field of supervised learning, studies on unsupervised learning have hardly been conducted, focusing on simple k-means clustering [4] and mean-shift clustering [36]. Because those methods in common struggle in capturing complex, non-convex cluster shapes, more complex clustering methods should be investigated. Therefore, we propose an homomorphic-encryption-friendly inference algorithm for support based clustering, which captures arbitrary clusters by

merging adjacent sub-clusters. Because most of the complex computations including merging sub-clusters are done in its training phase, its inference is simple yet effective, which suits the goal of the studies using homomorphic encryption.

4. **Differentially Private Mixture of Gaussians Clustering with Morse Theory** Despite the importance of unsupervised learning, there have been relatively few studies on differentially private clustering. Existing methods have been focused on k-means clustering [160] or mixture of Gaussians [138]] which cannot express complex, non-convex cluster shapes. To overcome the limitation, we introduce Morse theory and its associated dynamical system. The proposed method finds the transition equilibrium vectors of the dynamical system, and hierarchically merge pre-trained sub-clusters according to the density value of the transition equilibrium vectors. By using mixture of Gaussian as the density function, the proposed method can enhance the utility of existing differentially private clustering methods without incurring any additional privacy loss.

1.3 Organization of the Dissertation

The thesis is composed of 7 chapters. In Chapter 2, we give some background knowledge about homomorphic encryption and differential privacy. In Chapter 3, we propose an effective training of ridge regression with homomorphic encryption, and extend the method to parameter-free logistic regression in Chapter 4. In Chapter 5, we propose a homomorphic-encryption-friendly inference of support vector clustering. In Chapter 6, we propose a differentially private clustering method using Morse theory. Finally, in Chapter 7, we give concluding remarks and possible future research directions of this thesis.

Chapter 2

Preliminaries

This chapter presents a description of homomorphic encryption and differential privacy, which are the core concepts covered in this dissertation.

2.1 Homomorphic Encryption

Consider a situation in which a client has an input x and a server has a function f. The client wants to know the value of $f(x)$ without leaking the information regarding x, and the server does not want to give information regarding f. Notating the encryption of x as $\mathsf{Enc}(x)$, homomorphic encryption makes it possible to obtain $\mathsf{Enc}(f(x))$ using only $\mathsf{Enc}(x)$. Then, the client can decrypt $\mathsf{Enc}(f(x))$ to obtain the value of $f(x)$, but does not obtain any information about f, and the server does not know x without a secret key. Therefore, homomorphic encryption is the answer to the above problem. The security of an homomorphic encryption scheme is determined based on its underlying hardness assumption.

A general homomorphic encryption scheme has the following structure:

Definition 2.1. *[75, Definition 1] A homomorphic encryption scheme consists of four procedures:*

- $\mathsf{KeyGen}(1^\lambda, 1^\tau) \to (\mathsf{sKey}, \mathsf{pKey})$. *Input the security parameter* λ *and function-*

ality parameter τ, and output a secret key sKey *and a public key* pKey.

- $\mathsf{Encrypt}(\mathsf{pKey}, m) \rightarrow c$. *Input the public key and a plaintext m, and output the corresponding ciphertext c.*
- $\mathsf{Decrypt}(\mathsf{sKey}, c) \rightarrow m$. *Input the secret key and a ciphertext c, and output the corresponding plaintext m.*
- $\mathsf{Evaluate}(\mathsf{pKey}, f, \mathbf{c}) \rightarrow \mathbf{c}_f$. *For a vector of ciphertexts* $\mathbf{c}$ *for input of a circuit f, output a vector of ciphertexts* $\mathbf{c}$ *for output of f.*

The correctness of the scheme holds when for an arbitrary plaintext message m and its encryption c, $\mathsf{Evaluate}(\mathsf{pKey}, f, c)$ is equal to $\mathsf{Encrypt}(\mathsf{pKey}, f(m))$ where pKey is the public key generated from KeyGen [75]. Depending on f, homomorphic encryption has various applications from simple queries, such as information retrieval, to complex machine learning algorithms. The functionality parameter τ determines the bound of the depth of the circuit which can be evaluated using the scheme.

Cryptosystems for realizing homomorphic encryption have been under development for many years. Early schemes support only addition or multiplication, and are called partially homomorphic encryption. For example, the RSA cryptosystem [147] supports multiplication between ciphertexts, and the Paillier cryptosystem [134] supports the addition between ciphertexts and multiplication with plaintext. Until recently, the Paillier cryptosystem was applied to various machine learning models, but with the exception of simple inference models, it has a disadvantage in that all participants must be online because it requires multiple communication between participants.

In most schemes that support both addition and multiplication, noise is added to the ciphertext during the encryption process. The noise increases as it goes through a homomorphic operation. In particular, multiplication increases the noise more than addition. When the noise exceeds a certain threshold, the resulting ciphertext cannot be decrypted correctly, and a scheme that supports the number of operations to a certain extent is called a somewhat homomorphic encryption scheme. Among somewhat homomorphic encryption schemes, a scheme that can correctly evaluate an operation with a depth of up to a predetermined level is called a leveled homomorphic encryption scheme. By contrast, a fully homomorphic encryption scheme can evaluate any function having an arbitrary multiplicative depth.

Currently, the answer to evaluating an unbounded number of operations is bootstrapping. In 2009, Gentry's blueprint presented the first fully homomorphic encryption scheme with bootstrapping, which was based on an ideal lattice [61]. Bootstrapping refers to evaluating Decrypt for a ciphertext, resulting in a new ciphertext with reduced noise. However, because the decryption process is complex and nonlinear, bootstrapping is extremely costly. [5, 62, 76] suggested various optimization methods to improve the bootstrapping, but despite these efforts, bootstrapping is still the main bottleneck in many cases. In addition, because it requires encryption of the secret key, bootstrapping raises the problem of circular security, which is still an open question.

In a different direction from improvements to bootstrapping, studies have also been conducted to achieve more efficient leveled homomorphic encryption schemes. [19, 18] proposed different noise control methods through modulus switching and key switching, and in [63, 18, 158], among other studies, SIMD operations were made

possible by packing several plaintexts into a single ciphertext. Most of the recent schemes rely on the hardness of learning-with-errors, which becomes more efficient when applied to rings instead of integers. Later, [38, 39] proposed applying a more generalized learning-with-errors on a torus and developed a scheme using such an approach.

One aspect to consider when using homomorphic encryption is the use of non-polynomial functions. Non-polynomial functions such as exponential functions should be approximated as polynomial functions, and high-order polynomial operations are required to guarantee a high precision. Other functions need to be evaluated using iterative methods. For instance, division operations can be approximated using Goldschmidt's algorithm. Again, in this case, multiple iterations are required for high precision, and thus these operations are the main factor of increasing the multiplicative depth. Matrix inversion operations are more expensive, although they can be approximated using Schulz algorithm, which requires multiple matrix multiplications and requires strict conditions for numerical stability. Therefore, in [128, 8], a linear system-solution was converted into a least squares problem and approximated through a gradient descent.

2.2 Differential Privacy

Differential privacy is a privacy guarantee which ensures that that the outputs of a randomized algorithm are similar on similar input databases. For data owners, differential privacy guarantees that the outputs of the algorithm are similar regardless of which database their data resides in or not, thus reducing the risk of providing their data to an external database. The formal definition of differential privacy is as

follows:

Definition 2.2. *[54, Definition 2.4] A randomized algorithm $\mathcal{M}$ with domain $\mathcal{X}$ is (ε, δ)-differentially private if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for all two neighboring databases D, D' which differ in exactly one data sample,*

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq \exp(\varepsilon)\Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta. \tag{2.1}$$

If $\delta = 0$, it is said that $\mathcal{M}$ is ε-differentially private.

In the above definition, ε is a security parameter, and a smaller ε implies a higher level of security. δ is the probability of failure to guarantee privacy, and thus it should be set to a very small value. Generally, δ should be less than the inverse of any polynomial in the size of the database [54].

There have been various techniques proposed to make a function differentially private, which are called *mechanisms*. In this section, we introduce Laplace mechanism and Gaussian mechanism which are actually used in the study.

Definition 2.3. *[54, Definition 3.8] The l_p-sensitivity of a function $f : \mathcal{X} \to \mathbb{R}^k$ is:*

$$\Delta_p(f) = \max_{D,D' \in \mathcal{X}} ||f(x) - f(y)||_p, \tag{2.2}$$

where D, D' are two neighboring datasets.

Definition 2.4. *[54, Definition 3.2] The Laplace distribution (centered at 0) with scale b is the distribution with probability density function:*

$$\text{Lap}(x; b) = \frac{1}{2b}\exp(-\frac{|x|}{b}). \tag{2.3}$$

Definition 2.5. *[54, Definition 3.2] Given any function $f : \mathcal{X} \to \mathbb{R}^k$, the Laplace mechanism is defined as:*

$$\mathcal{M}_L(x, f(\cdot), \varepsilon) = f(x) + (Z_1, \ldots, Z_k) \tag{2.4}$$

where Z_i are i.i.d. random variables drawn from $\mathrm{Lap}(\Delta_1(f)/\varepsilon)$.

Remark 2.1. *[54, Theorem 3.6] The Laplace mechanism guarantees ε-differential privacy.*

Similar to the Laplace mechanism, the Gaussian mechanism with parameter σ is simply adding zero-mean Gaussian noise $Z \sim \mathcal{N}(0, \sigma^2 I)$ to a function f, where σ is scaled with the l_2 sensitivity of f.

Remark 2.2. *[54, Theorem 3.22] For any $\varepsilon \in (0, 1)$, the Gaussian mechanism with $\sigma^2 = 2\log(1.25/\delta)(\Delta_2(f))^2/\varepsilon^2$ is (ε, δ)-differentially private.*

There are several good properties of differential privacy which makes it possible to guarantee differential privacy for complex machine learning algorithms. The first is *Post-processing* property, which ensures that composition of a data-independent mapping with a differentially private algorithm does not pose any additional privacy threats.

Remark 2.3. *[54, Proposition 2.1] Let $\mathcal{M} : \mathcal{X} \to R$ be an (ε, δ)-differentially private randomized algorithm, $f : R \to R'$ be an arbitrary randomized mapping. Then $f \circ \mathcal{M} : \mathcal{X} \to R'$ is (ε, δ)-differentially private.*

Another property, composition tells how the privacy guarantee changes when multiple differentially private algorithms are applied. Parallel composition ensures

that the privacy loss does not grow when the algorithm's inputs are isolated from each other.

Remark 2.4 (Parallel Composition)**.** *Let the dataset D is partitioned by disjoint subset D_i for $i = 1, 2, \ldots, n$ and let $\mathcal{M}_i$ be $(\varepsilon_i, \delta_i)$-differentially private algorithm which takes D_i as input. Then, releasing $\mathcal{M}_i$s is $(\max_i \varepsilon_i, \max_i \delta_i)$-differentially private.*

In general, using multiple differentially private mechanisms increases the privacy loss, which is estimated through various composition theorems. To obtain a rather tight upper bound of the privacy loss, we used the concept of zero-centered differential privacy [21], which is another notion of differential privacy and compatible with the original definition. Below we briefly introduce some properties of zero-centered differential privacy.

Remark 2.5. *[21, Proposition 1.3] If $\mathcal{M}$ satisfies ρ-zero-centered differential privacy, then $\mathcal{M}$ satisfies $(\rho + 2\sqrt{\rho \log(1/\delta)}, \delta)$-differential privacy for any $\delta > 0$.*

Remark 2.6. *[21, Proposition 1.4] If $\mathcal{M}$ satisfies ε-differential privacy, then $\mathcal{M}$ satisfies $\varepsilon^2/2$-zero-centered differential privacy.*

Remark 2.7. *[21, Proposition 1.6] Let function $f : \mathcal{X} \to \mathbb{R}$ has a sensitivity Δ. Then on an input x, releasing a sample from $\mathcal{N}(f(x), \sigma^2)$ satisfies $(\Delta^2/2\sigma^2)$-zero-centered differential privacy.*

Remark 2.8. *[21, Lemma 1.8] Let randomized algorithms $\mathcal{M}_1$ and $\mathcal{M}_\in$ satisfy ρ_1-zero-centered differential privacy and ρ_2-zero-centered differential privacy, respectively. Then the mapping $\mathcal{M}_{12} = (\mathcal{M}_1, \mathcal{M}_2)$ satisfies $(\rho_1 + \rho_2)$-zero-centered differential privacy.*

Chapter 3

Efficient Homomorphic Encryption Framework for Ridge Regression

3.1 Problem Statement

During the past few years, machine learning has shown remarkable achievements in almost every sector, including autonomous driving, marketing and finance [89, 122, 48]. The success of machine learning is due to the explosion in the number of published data, as well as advances in computational and storage capabilities represented by GPUs. However, individual data contain private information, which has raised concerns regarding the infringement of data privacy. Even if the individuals have agreed to provide such information, an unwanted level of information breach may occur if the database is attacked. Representatively, owing to the Facebook-Cambridge Analytica Data Breach in 2018, a number of people became aware of the seriousness of information leaks. Before and after, legislation such as the European General Data Protection Regulation (GDPR) [169] was enacted to protect personal privacy and information, which commonly mandates reliable protection in the use of data by third parties.

Therefore, applying security techniques in data publishing is no longer optional, but essential. Among privacy-preserving techniques, homomorphic encryption (HE)

ensures data security without damaging the information contained in the data. HE enables computation on encrypted data, and it is possible to translate arbitrary machine learning techniques into a combination of homomorphic operations. HE guarantees the highest level of security as long as the scheme has not cracked, and recently proposed LWE-based HE schemes are known to be resistant against quantum attacks. The biggest factors limiting the utilization of HE in machine learning are time and storage efficiency. Currently, a state-of-the-art neural-network-training model requires 8 days for training with the encrypted MNIST dataset, which can be achieved in a few minutes with plaintext [118]. As a result, only the evaluation phase has been implemented in many studies; however, given that the main purpose of data collection is for training, model training should be further studied.

Ridge regression is an essential building block for various machine learning techniques, which models the linear relationship between input variables and a dependent variable. The most expensive part of a ridge regression learning process is a matrix inversion operation that solves a linear system, which becomes far more difficult for encrypted data. [8] and [128] have proposed methods which obtains an approximate solution through a gradient descent instead of directly solving the linear system. Their methods are computationally costly, and setting an appropriate learning rate remains an open problem.

It is widely accepted that security for the sensitive variables is more important than security for other variables. [24] and [74] proposed data transformation methods to prevent inference attack for sensitive attributes, and [115] proposed a the method for dealing with multiple numerical sensitive attributes. [163, 60] selectively encrypted the sensitive variables to provide a trade-off among privacy, time,

and storage efficiency. Another example that has evident importance for sensitive variables is fair machine learning (FML), which has recently attracted significant interest [58, 79, 112]. In FML, information that can result in discrimination, such as gender or race, is regarded as a sensitive attribute.

In this study, we propose a system to securely perform ridge regression in which only sensitive variables are encrypted using HE. First, we present a novel efficient algorithm of ridge regression for two sensitive variables. Then, we propose a defense method which can protect sensitive variables from a machine learning service provider (MLSP) who conducts outsourced computations for training ML models.

The main contributions of this study are as follows.

- Our method can be implemented with any HE scheme. Although unexpected challenges might occur when using special HE schemes in practice, because our algorithm only uses the general framework of HE that is shared by most HE schemes, our method operate effectively regardless of the scheme used and can be easily updated by an update version of HE scheme.

- We suggest a new perspective to an efficiency-security trade-off, which is currently the most notable issue in the application of HE [94]. When each ciphertext contains a different column, the complexity of our efficient ridge regression method is $O(d \log n)$, whereas the complexity of a method encrypting the entire data is $O(d^2 \log n)$. The experimental results using 11 real-world datasets support the claim.

- Our proposed defense system makes our method secure against attribute inference attacks on sensitive variables. The method can be applied to continuous

variables as well as categorical variables, and the experimental results show that our defense method effectively defend against the attacks while little affecting the performance of ridge regression.

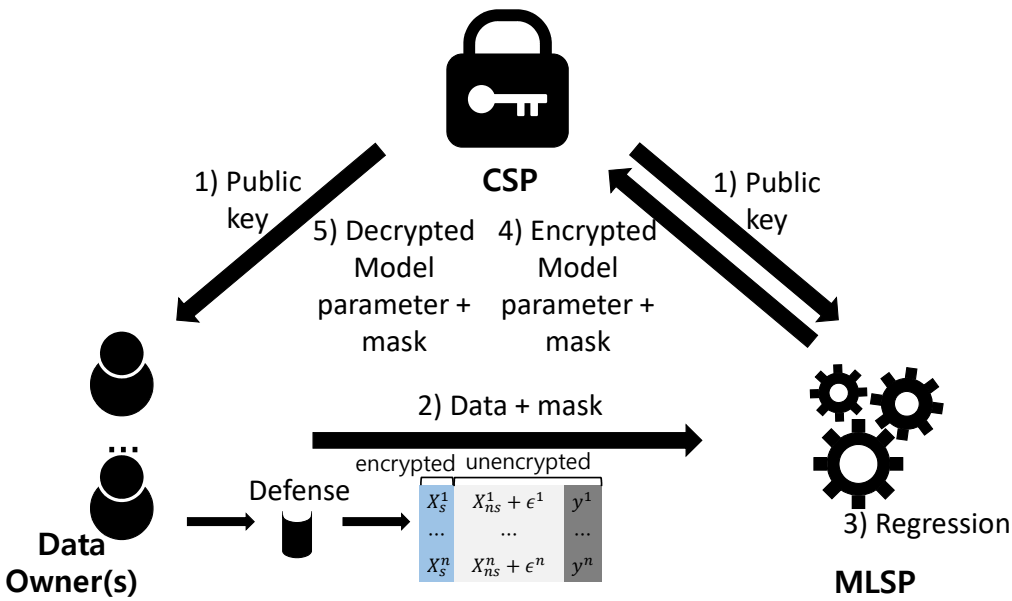


Figure 3.1: Description of overall system

3.2 Framework

Our system resembles a three-party model, which is widely used in homomorphic encryption machine learning approaches. It consists of data owners, a machine learning service provider (MLSP), and a crypto-service provider (CSP). The CSP first generates a public key and a secret key for the scheme and publishes the public key to the MLSP and data owners. Data owners encrypt their sensitive attributes with the public key and send the encrypted sensitive attributes and other (unencrypted) variables to the MLSP. The MSLP then conducts privacy-preserving ridge regression using the data and sends the encrypted trained model parameters to the CSP.

Finally, the CSP decrypts the result using the secret key and sends the result back to the data owners.

We assume that the MLSP and the CSP are honest-but-curious and do not collude with each other, as assumed in [144, 128].

Therefore, they follow the procedure correctly, although they can try to obtain hidden information from consumers by using the information given to them. The security goal under this assumption is defined as in [139].

Definition 3.1. *[139, Definition 1] The proposed protocol is defined to be* secure *if the following holds.*

- *(Privacy of sensitive user information) Neither the CSP nor the MLSP gets any information for the sensitive variables except for their respective outputs.*
- *(Model secrecy) The output of the training is not disclosed to the CSP or data owners.*
- *(Minimal disclosure of non-sensitive attributes) None of the user data, including non-sensitive variables, are exposed to the CSP.*

With an appropriate security parameter, the privacy of the proposed framework can be guaranteed directly following the proof of [139]. However, in this study we investigate another possible privacy breach, in which the MLSP infers the values of sensitive variables using unencrypted variables with the help of external knowledge. Such attacks have been referred to as linking attacks or attribute inference attacks [105, 59]. [91] proposed a defense method based on the evasion attack, which is a kind of an adversarial attack. However, their method can deal with only discrete sensitive variables, and it requires a separate optimization for each data point. Therefore,

in this study, we propose a novel defense algorithm against attribute inference attack, which compensates for the shortcomings of [91]. Because the defense must be performed before data owners transfer the data to the MLSP, the existence of a reliable defender is additionally required if data owners themselves cannot perform the defense.

As another security concern, the CSP can directly decrypt the final result using the secret key. However, this type of attack can be easily prevented with a slightly higher communication cost. Data owners first determine the values for a random vector with the same length as the final result, which is called a mask. They then encrypt the mask and send it to the MLSP with the data. After the computation of the MLSP is applied, the MLSP adds the mask to the final encrypted result. Because the CSP does not know the distribution of the values for the mask, no information can be obtained from the decrypted result. The overall protocol of our system is shown in Figure 3.1.

To operate between ciphertexts, the number of plaintext slots for each ciphertext must be the same. Because the number of the data may vary for each owner, data owners should determine the size of the plaintext slot in advance. That is, if the number of data points that a data owner has is r and the size of the plaintext slot is N, the data owner should split each of the columns into $\lceil r/N \rceil$ pieces, and 0-pad the last piece such that its length is N. We note that the setup is no different than having a single data owner owning multiple pieces of data. Therefore, in the next section, our method is described under the assumption that one data owner owns all of the data.

3.3 Proposed Method

3.3.1 Regression with one Encrypted Sensitive Variable

We start with the following multivariate linear regression with one dependent variable Y, p- independent *non-private* variables $X_1, X_2, \ldots, X_p$, and one *sensitive variable* X_s:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \beta_s X_s + \varepsilon \tag{3.1}$$

where $\boldsymbol{\beta}_I = (\beta_0, \beta_1, ..., \beta_p, \beta_s)^T$ consists of regression coefficients to be estimated and ε is error term. For $i = 1, ...n$, $(x_{i1}, \ldots, x_{ip}, x_{is}, y_i)$ denotes the i-th observations of $X_1, X_2, \ldots, X_p$, X_s, and Y. We assume that $n > p + 1$, which means that there are more observations than the number of independent variables.

Encryption of a sensitive variable Let $h_i = h(x_{is})$ be the fully homomorphic encryption of a sensitive variable x_{is}, then the data transferred to the server is $(x_{i1}, \ldots, x_{ip}, h(x_{is}), y_i)$ for $i = 1, ...n$. For the convenience, from now on, operations between a plaintext and a ciphertext, or between ciphertexts, are denoted as those between plaintexts.

Using centered inputs by replacing each x_{ij} with $x_{ij} - \bar{x}_j$, each $h(x_{is})$ with $h(x_{is}) - \bar{h}_s$, and estimating β_0 by $\bar{y} = \sum_{i=1}^{n} y_i$, (3.1) becomes a regression model

without intercept as

$$
\begin{aligned}
& y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \beta_s h(x_{is}) + \varepsilon_i, \quad i = 1, ..., n \\
& \boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \\
& = \underbrace{\begin{bmatrix} x_{11} & \cdots & x_{1p} & h(x_{1s}) \\ \vdots & \ddots & \vdots & \vdots \\ x_{i1} & \cdots & x_{ip} & h(x_{is}) \\ \vdots & \ddots & \vdots & \vdots \\ x_{n1} & \cdots & x_{np} & h(x_{ns}) \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} \beta_1 \\ \vdots \\ \beta_p \\ \beta_s \end{bmatrix}}_{\beta} + \underbrace{\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}}_{\varepsilon}
\end{aligned} \tag{3.2}
$$

Ridge estimate without matrix inverse By adding a regularization term to an error function in order to control over-fitting, the total error function to be minimized takes the form

$$
\begin{aligned}
\mathcal{E}(\boldsymbol{\beta}) =& \mathcal{E}_D(\boldsymbol{\beta}) + \mathcal{E}_W(\boldsymbol{\beta}) \\
=& \frac{1}{2}\sum_{i=1}^{N}(y_i - (\beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \beta_s h(x_{is})))^2 + \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\beta}.
\end{aligned} \tag{3.3}
$$

The ridge regression solutions are then shown to be

$$
\hat{\boldsymbol{\beta}}_{RLS} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p+1})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{3.4}
$$

where $\boldsymbol{I}_{p+1}$ is a $(p+1) \times (p+1)$-identity matrix. Note that the intercept β_0 is not regularized because of centering of the variables. The ridge estimate is therefore

$$
\hat{\boldsymbol{f}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p+1})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{y} \tag{3.5}
$$

where $\boldsymbol{X}\boldsymbol{X}^T$ is an $n \times n$ matrix.

The last equality holds because $\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_n) = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p+1})\boldsymbol{X}^T$, and therefore $(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p+1})^{-1}\boldsymbol{X}^T = \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_n)^{-1}$.

Since

$$\boldsymbol{X}\boldsymbol{X}^T = \sum_{i=1}^{p} \boldsymbol{x}_i \boldsymbol{x}_i^T + \boldsymbol{h}_s \boldsymbol{h}_s^T = \boldsymbol{X}_{(-s)}\boldsymbol{X}_{(-s)}^T + \boldsymbol{h}_s \boldsymbol{h}_s^T \tag{3.6}$$

where $\boldsymbol{h}_s = (h(x_{1s}), ..., h(x_{ns}))^T$ and $\boldsymbol{X}_{(-s)}$ is the other part of $\boldsymbol{X}$, by Sherman-Woodbury inversion formula, we have

$$\begin{aligned}
(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_n)^{-1} &= (\boldsymbol{X}_{(-s)}\boldsymbol{X}_{(-s)}^T + \lambda \boldsymbol{I}_n + \boldsymbol{h}_s \boldsymbol{h}_s^T)^{-1} \\
= (\boldsymbol{X}_{(-s)}\boldsymbol{X}_{(-s)}^T + \lambda \boldsymbol{I}_n)^{-1} &- \frac{(\boldsymbol{X}_{(-s)}\boldsymbol{X}_{(-s)}^T + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{h}_s \boldsymbol{h}_s^T (\boldsymbol{X}_{(-s)}\boldsymbol{X}_{(-s)}^T + \lambda \boldsymbol{I}_n)^{-1}}{1 + \boldsymbol{h}_s^T (\boldsymbol{X}_{(-s)}\boldsymbol{X}_{(-s)}^T + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{h}_s} \\
&= \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{h}_s \boldsymbol{h}_s^T \boldsymbol{A}^{-1}}{1 + \boldsymbol{h}_s^T \boldsymbol{A}^{-1}\boldsymbol{h}_s}
\end{aligned} \tag{3.7}$$

where $\boldsymbol{A} = \boldsymbol{X}_{(-s)}\boldsymbol{X}_{(-s)}^T + \lambda \boldsymbol{I}_n$. Using the singular vector decomposition (SVD) of $\boldsymbol{X}_{(-s)} = U\Sigma V^T$ where $U \in \Re^{n \times n}$ and $V \in \Re^{p \times p}$ are orothogonal matrices and $\Sigma \in \Re^{n \times p}$ are diagonal matrix with diagonal entries $\sigma_1 \geq .... \geq \sigma_p$, we have

$$\boldsymbol{A}^{-1} = (\boldsymbol{X}_{(-s)}\boldsymbol{X}_{(-s)}^T + \lambda \boldsymbol{I}_n)^{-1} = (U\Sigma\Sigma^T U^T + \lambda \boldsymbol{I}_n)^{-1} = U(\Sigma\Sigma^T + \lambda \boldsymbol{I}_n)^{-1}U^T \tag{3.8}$$

and letting $\sigma_{p+1} = \cdots = \sigma_n = 0$, we define the following terms

$$\xi = \boldsymbol{h}_s^T \boldsymbol{A}^{-1} \boldsymbol{h}_s = \boldsymbol{h}_s^T U(\Sigma\Sigma^T + \lambda \boldsymbol{I}_n)^{-1} U^T \boldsymbol{h}_s = \sum_{j=1}^{n} \boldsymbol{h}_s^T \boldsymbol{u}_j \frac{1}{\sigma_j^2 + \lambda} \boldsymbol{u}_j^T \boldsymbol{h}_s = \sum_{j=1}^{n} \frac{(\boldsymbol{h}_s^T \boldsymbol{u}_j)^2}{\sigma_j^2 + \lambda}$$

$$\eta = \boldsymbol{h}_s^T \boldsymbol{A}^{-1} \boldsymbol{y} = \boldsymbol{h}_s^T U(\Sigma\Sigma^T + \lambda \boldsymbol{I}_n)^{-1} U^T \boldsymbol{y} = \sum_{j=1}^{n} \frac{(\boldsymbol{h}_s^T \boldsymbol{u}_j)(\boldsymbol{u}_j^T \boldsymbol{y})}{\sigma_j^2 + \lambda}. \tag{3.9}$$

Then the ridge estimate becomes

$$\begin{aligned}
\hat{\boldsymbol{f}} =& \boldsymbol{X}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{y} = (\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_n - \lambda \boldsymbol{I}_n)(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{y} \\
=& \boldsymbol{y} - \lambda(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_n)^{-1} \boldsymbol{y} = \boldsymbol{y} - \lambda \boldsymbol{A}^{-1} \boldsymbol{y} + \lambda \frac{\boldsymbol{A}^{-1} \boldsymbol{h}_s \boldsymbol{h}_s^T \boldsymbol{A}^{-1}}{1 + \boldsymbol{h}_s^T \boldsymbol{A}^{-1} \boldsymbol{h}_s} \boldsymbol{y} \\
=& \boldsymbol{y} - \lambda \boldsymbol{A}^{-1} \boldsymbol{y} + \frac{\lambda \eta}{1 + \xi} \boldsymbol{A}^{-1} \boldsymbol{h}_s \\
=& \boldsymbol{y} - \lambda U(\Sigma\Sigma^T + \lambda \boldsymbol{I}_n)^{-1} U^T \boldsymbol{y} + \frac{\lambda \eta}{1 + \xi} U(\Sigma\Sigma^T + \lambda \boldsymbol{I}_n)^{-1} U^T \boldsymbol{h}_s \\
=& \boldsymbol{y} - \sum_{j=1}^{n} \frac{\lambda \boldsymbol{u}_j (\boldsymbol{u}_j^T \boldsymbol{y})}{\sigma_j^2 + \lambda} + \frac{\lambda \eta}{1 + \xi} \sum_{j=1}^{n} \frac{\boldsymbol{u}_j (\boldsymbol{u}_j^T \boldsymbol{h}_s)}{\sigma_j^2 + \lambda} \\
=& \sum_{j=1}^{n} \frac{\sigma_j^2}{\sigma_j^2 + \lambda} \boldsymbol{u}_j (\boldsymbol{u}_j^T \boldsymbol{y}) + \frac{\lambda \eta}{1 + \xi} \sum_{j=1}^{n} \frac{\boldsymbol{u}_j (\boldsymbol{u}_j^T \boldsymbol{h}_s)}{\sigma_j^2 + \lambda}
\end{aligned} \tag{3.10}$$

where $\boldsymbol{u}_j$ are the columns of $U \in \Re^{n \times n}$ and $\sum_{j=1}^{n} \boldsymbol{u}_j \boldsymbol{u}_j^T = \boldsymbol{I}_n$.

Fast ridge estimate The above computation involves n summations. To simplify the computations, we notice that $U = [U_1, U_2] \in \Re^{n \times n}$ where $U_1 = [\boldsymbol{u}_1, ..., \boldsymbol{u}_p] \in \Re^{n \times p}$ and $U_2 = [\boldsymbol{u}_{p+1}, ..., \boldsymbol{u}_n] \in \Re^{n \times (n-p)}$ and $U_1 \perp U_2$. Using the reduced SVD and the fact that $\sigma_{p+1} = \cdots = \sigma_n = 0$, we have the freedom to choose U_2 as long as $U_1 \perp U_2$. Therefore, we choose $\boldsymbol{u}_{p+1}$ in such a way that $\boldsymbol{u}_j$ are orthogonal to $\boldsymbol{h}_s$ for all $j = p+2, ..., n$ by letting $\boldsymbol{u}_{p+1}$ be the complement of the orthogonal projection

of $\boldsymbol{h}_s$ onto U_1 as follows:

$$\hat{\boldsymbol{u}}_{p+1} = (\boldsymbol{I}_n - P_{U_1})\boldsymbol{h}_s = (\boldsymbol{I}_n - \sum_{i=1}^{p} \boldsymbol{u}_i \boldsymbol{u}_i^T)\boldsymbol{h}_s = \boldsymbol{h}_s - \sum_{i=1}^{p} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{h}_s)$$
$$\boldsymbol{u}_{p+1} = \hat{\boldsymbol{u}}_{p+1}/\|\hat{\boldsymbol{u}}_{p+1}\|, \quad \boldsymbol{u}_{p+1}(\boldsymbol{u}_{p+1}^T \boldsymbol{h}_s) = \boldsymbol{h}_s - \sum_{i=1}^{p} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{h}_s). \tag{3.11}$$

Then ξ and η can be simplified as

$$\begin{aligned}
\eta = \sum_{j=1}^{p+1} \frac{\boldsymbol{h}_s^T \boldsymbol{u}_j}{\sigma_j^2 + \lambda} \boldsymbol{u}_j^T \boldsymbol{y} &= \sum_{j=1}^{p} \frac{\boldsymbol{h}_s^T \boldsymbol{u}_j}{\sigma_j^2 + \lambda} \boldsymbol{u}_j^T \boldsymbol{y} + \frac{1}{\lambda} \boldsymbol{y}^T \boldsymbol{u}_{p+1}(\boldsymbol{u}_{p+1}^T \boldsymbol{h}_s) \\
&= \sum_{j=1}^{p} \frac{(\boldsymbol{h}_s^T \boldsymbol{u}_j)(\boldsymbol{u}_j^T \boldsymbol{y})}{\sigma_j^2 + \lambda} + \frac{1}{\lambda} \boldsymbol{y}^T (\boldsymbol{h}_s - \sum_{j=1}^{p} \boldsymbol{u}_j(\boldsymbol{u}_j^T \boldsymbol{h}_s)) \\
&= \frac{1}{\lambda} \left[- \sum_{j=1}^{p} \frac{\sigma_j^2 (\boldsymbol{h}_s^T \boldsymbol{u}_j)(\boldsymbol{u}_j^T \boldsymbol{y})}{\sigma_j^2 + \lambda} + \boldsymbol{y}^T \boldsymbol{h}_s \right] \\
\xi = \sum_{j=1}^{p+1} \frac{(\boldsymbol{h}_s^T \boldsymbol{u}_j)^2}{\sigma_j^2 + \lambda} &= \sum_{j=1}^{p} \frac{(\boldsymbol{h}_s^T \boldsymbol{u}_j)^2}{\sigma_j^2 + \lambda} + \frac{1}{\lambda} \boldsymbol{h}_s^T \boldsymbol{u}_{p+1}(\boldsymbol{u}_{p+1}^T \boldsymbol{h}_s) \\
&= \sum_{j=1}^{p} \frac{(\boldsymbol{h}_s^T \boldsymbol{u}_j)^2}{\sigma_j^2 + \lambda} + \frac{1}{\lambda} \boldsymbol{h}_s^T (\boldsymbol{h}_s - \sum_{j=1}^{p} \boldsymbol{u}_j(\boldsymbol{u}_j^T \boldsymbol{h}_s)) \\
&= \frac{1}{\lambda} \left[- \sum_{j=1}^{p} \frac{\sigma_j^2 (\boldsymbol{h}_s^T \boldsymbol{u}_j)^2}{\sigma_j^2 + \lambda} + \boldsymbol{h}_s^T \boldsymbol{h}_s \right].
\end{aligned} \tag{3.12}$$

Therefore the ridge estimate can be further simplified as

$$
\begin{aligned}
\hat{\boldsymbol{f}} &= \sum_{i=1}^{p} \frac{\sigma_i^2}{\sigma_i^2+\lambda} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{y}) + \frac{\lambda\eta}{1+\xi} \sum_{j=1}^{p+1} \frac{\boldsymbol{u}_j(\boldsymbol{u}_j^T \boldsymbol{h}_s)}{\sigma_j^2+\lambda} \\
&= \sum_{i=1}^{p} \frac{\sigma_i^2}{\sigma_i^2+\lambda} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{y}) + \frac{\lambda\eta}{1+\xi} \left[\sum_{j=1}^{p} \frac{\boldsymbol{u}_j(\boldsymbol{u}_j^T \boldsymbol{h}_s)}{\sigma_j^2+\lambda} + \frac{1}{\lambda}(\boldsymbol{h}_s - \sum_{i=1}^{p} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{h}_s)) \right] \\
&= \sum_{i=1}^{p} \boldsymbol{u}_i \frac{\sigma_i^2}{\sigma_i^2+\lambda} \boldsymbol{u}_i^T \boldsymbol{y} - \frac{\eta}{1+\xi} \sum_{i=1}^{p} \boldsymbol{u}_i \frac{\sigma_i^2(\boldsymbol{u}_i^T \boldsymbol{h}_s)}{\sigma_i^2+\lambda} + \frac{\eta}{1+\xi} \boldsymbol{h}_s
\end{aligned} \tag{3.13}
$$

which involves only p summations.

3.3.2 Regression with two Encrypted Sensitive Variables

Depending on the characteristics of data, it may be necessary to protect multiple attributes. For example, when gender and race are included in the same dataset, both may need to be treated as sensitive attributes. Therefore, we extend the previous result for the case where there are two sensitive variables.

Similar to (3.6),

$$
\boldsymbol{X}\boldsymbol{X}^T = \sum_{i=1}^{p} \boldsymbol{x}_i \boldsymbol{x}_i^T + \boldsymbol{h}\boldsymbol{h}^T + \boldsymbol{g}\boldsymbol{g}^T = \boldsymbol{X}_{(-s)} \boldsymbol{X}_{(-s)}^T + \boldsymbol{h}\boldsymbol{h}^T + \boldsymbol{g}\boldsymbol{g}^T \tag{3.14}
$$

where $\boldsymbol{h} = (h(x_{1s1}), ..., h(x_{ns1}))^T \in \Re^n$ and $\boldsymbol{g} = (g(x_{1s1}), ..., g(x_{ns1}))^T \in \Re^n$ are two encrypted sensitive variables.

Then following the similar process as 3.3.1, the ridge solution becomes

$$\hat{\boldsymbol{f}} = \sum_{j=1}^{n} \frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{u}_j(\boldsymbol{u}_j^T\boldsymbol{y}) + \frac{\lambda(\eta_1+\xi_{22}\eta_1-\xi_{12}\eta_2)}{(1+\xi_{11})(1+\xi_{22})-\xi_{12}^2}\sum_{j=1}^{n}\frac{1}{\sigma_j^2+\lambda}\boldsymbol{u}_j(\boldsymbol{u}_j^T\boldsymbol{h}) + \frac{\lambda(\eta_2+\xi_{11}\eta_2-\xi_{12}\eta_1)}{(1+\xi_{11})(1+\xi_{22})-\xi_{12}^2}\sum_{j=1}^{n}\frac{1}{\sigma_j^2+\lambda}\boldsymbol{u}_j(\boldsymbol{u}_j^T\boldsymbol{g}) \tag{3.15}$$

where

$$\begin{aligned}
\xi_{11} = \boldsymbol{h}^T\boldsymbol{A}^{-1}\boldsymbol{h} &= \boldsymbol{h}^T U(\Sigma\Sigma^T+\lambda\boldsymbol{I}_n)^{-1}U^T\boldsymbol{h} = \sum_{j=1}^{n}\boldsymbol{h}^T\boldsymbol{u}_j\frac{1}{\sigma_j^2+\lambda}\boldsymbol{u}_j^T\boldsymbol{h} = \sum_{j=1}^{n}\frac{(\boldsymbol{h}^T\boldsymbol{u}_j)^2}{\sigma_j^2+\lambda}\\
\xi_{12} &= \boldsymbol{h}^T\boldsymbol{A}^{-1}\boldsymbol{g} = \sum_{j=1}^{n}\boldsymbol{h}^T\boldsymbol{u}_j\frac{1}{\sigma_j^2+\lambda}\boldsymbol{u}_j^T\boldsymbol{g} = \sum_{j=1}^{n}\frac{(\boldsymbol{h}^T\boldsymbol{u}_j)(\boldsymbol{g}^T\boldsymbol{u}_j)}{\sigma_j^2+\lambda}\\
\xi_{22} &= \boldsymbol{g}^T\boldsymbol{A}^{-1}\boldsymbol{g} = \sum_{j=1}^{n}\boldsymbol{g}^T\boldsymbol{u}_j\frac{1}{\sigma_j^2+\lambda}\boldsymbol{u}_j^T\boldsymbol{g} = \sum_{j=1}^{n}\frac{(\boldsymbol{g}^T\boldsymbol{u}_j)^2}{\sigma_j^2+\lambda}\\
\eta_1 &= \boldsymbol{h}^T\boldsymbol{A}^{-1}\boldsymbol{y} = \boldsymbol{h}^T U(\Sigma\Sigma^T+\lambda\boldsymbol{I}_n)^{-1}U^T\boldsymbol{y} = \sum_{j=1}^{n}\frac{(\boldsymbol{h}^T\boldsymbol{u}_j)(\boldsymbol{u}_j^T\boldsymbol{y})}{\sigma_j^2+\lambda}\\
\eta_2 &= \boldsymbol{g}^T\boldsymbol{A}^{-1}\boldsymbol{y} = \boldsymbol{g}^T U(\Sigma\Sigma^T+\lambda\boldsymbol{I}_n)^{-1}U^T\boldsymbol{y} = \sum_{j=1}^{n}\frac{(\boldsymbol{g}^T\boldsymbol{u}_j)(\boldsymbol{u}_j^T\boldsymbol{y})}{\sigma_j^2+\lambda}.
\end{aligned} \tag{3.16}$$

Fast ridge estimate Extending the results from 3.3.1, we have the freedom to choose U_2 as long as $U_1 \perp U_2$ in such a way that the orthogonalization process is applied to $\boldsymbol{h}$ first and $\boldsymbol{g}$ next.

- $\boldsymbol{u}_j$ are orthogonal to $\boldsymbol{h}$ for all $j = p+2, ..., n$ and
- $\boldsymbol{u}_j$ are orthogonal to $\boldsymbol{g}$ for all $j = p+3, ..., n$ and
- $\boldsymbol{u}_{p+1}$ and $\boldsymbol{u}_{p+2}$ are the complements of the orthogonal projections of $\boldsymbol{h}$ and $\boldsymbol{g}$ onto U_1 and $[U_1, \boldsymbol{u}_{p+1}]$, respectively and $\boldsymbol{u}_{p+1} \perp \boldsymbol{u}_{p+2}$.

That is,

$$\begin{aligned}
\hat{\boldsymbol{u}}_{p+1} =& \boldsymbol{h} - \sum_{i=1}^{p} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{h}) \\
\boldsymbol{u}_{p+1} =& \hat{\boldsymbol{u}}_{p+1}/\|\hat{\boldsymbol{u}}_{p+1}\|, \quad \boldsymbol{u}_{p+1}(\boldsymbol{u}_{p+1}^T \boldsymbol{h}) = \boldsymbol{h} - \sum_{i=1}^{p} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{h}) \\
\hat{\boldsymbol{u}}_{p+2} =& \boldsymbol{g} - \sum_{i=1}^{p+1} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{g}), \quad \boldsymbol{u}_{p+2} = \hat{\boldsymbol{u}}_{p+2}/\|\hat{\boldsymbol{u}}_{p+2}\|.
\end{aligned}$$

Then following (3.12), ξ_{11}, ξ_{12} and η_1 can be simplified as

$$\begin{aligned}
\xi_{11} =& \sum_{j=1}^{n} \frac{(\boldsymbol{h}^T \boldsymbol{u}_j)^2}{\sigma_j^2 + \lambda} = \sum_{j=1}^{p} \frac{(\boldsymbol{h}^T \boldsymbol{u}_j)^2}{\sigma_j^2 + \lambda} + \frac{1}{\lambda} \boldsymbol{h}^T \boldsymbol{u}_{p+1}(\boldsymbol{u}_{p+1}^T \boldsymbol{h}) \\
=& \frac{1}{\lambda} \left[- \sum_{j=1}^{p} \frac{\sigma_j^2 (\boldsymbol{h}^T \boldsymbol{u}_j)^2}{\sigma_j^2 + \lambda} + \boldsymbol{h}^T \boldsymbol{h} \right] \\
\xi_{12} =& \sum_{j=1}^{n} \frac{(\boldsymbol{h}^T \boldsymbol{u}_j)(\boldsymbol{g}^T \boldsymbol{u}_j)}{\sigma_j^2 + \lambda} = \sum_{j=1}^{p} \frac{\boldsymbol{g}^T \boldsymbol{u}_j (\boldsymbol{u}_j^T \boldsymbol{h})}{\sigma_j^2 + \lambda} + \frac{1}{\lambda} \boldsymbol{g}^T \boldsymbol{u}_{p+1}(\boldsymbol{u}_{p+1}^T \boldsymbol{h}) \\
=& \frac{1}{\lambda} \left[- \sum_{j=1}^{p} \frac{\sigma_j^2 (\boldsymbol{h}^T \boldsymbol{u}_j)(\boldsymbol{g}^T \boldsymbol{u}_j)}{\sigma_j^2 + \lambda} + \boldsymbol{g}^T \boldsymbol{h} \right] \\
\eta_1 =& \sum_{j=1}^{n} \frac{(\boldsymbol{h}^T \boldsymbol{u}_j)(\boldsymbol{u}_j^T \boldsymbol{y})}{\sigma_j^2 + \lambda} = \sum_{j=1}^{p} \frac{(\boldsymbol{y}^T \boldsymbol{u}_j)(\boldsymbol{u}_j^T \boldsymbol{h})}{\sigma_j^2 + \lambda} + \frac{1}{\lambda} \boldsymbol{y}^T (\boldsymbol{h} - \sum_{j=1}^{p} \boldsymbol{u}_j (\boldsymbol{u}_j^T \boldsymbol{h})) \\
=& \frac{1}{\lambda} \left[- \sum_{j=1}^{p} \frac{\sigma_j^2 (\boldsymbol{h}^T \boldsymbol{u}_j)(\boldsymbol{y}^T \boldsymbol{u}_j)}{\sigma_j^2 + \lambda} + \boldsymbol{y}^T \boldsymbol{h} \right]
\end{aligned} \tag{3.17}$$

At a glance, the calculation of ξ_{22}, η_2 seems to be more complicated because it includes $\boldsymbol{u}_{p+2}$, which is calculated using $\boldsymbol{u}_{p+1}$. However, ξ_{22}, η_2 can be further

simplified without using $\boldsymbol{u}_{p+2}$. To do this, we observe that without loss of generality, we can apply the orthogonalization process to $\boldsymbol{g}$ first and $\boldsymbol{h}$ next as follows.

- $\boldsymbol{u}_j$ are orthogonal to $\boldsymbol{g}$ for all $j = p+2, ..., n$ and
- $\boldsymbol{u}_j$ are orthogonal to $\boldsymbol{h}$ for all $j = p+3, ..., n$ and
- $\boldsymbol{u}_{p+1}$ and $\boldsymbol{u}_{p+2}$ are the orthogonal projections of $\boldsymbol{g}$ and $\boldsymbol{h}$ onto U_1 and $[U_1, \boldsymbol{u}_{p+1}]$, respectively and $\boldsymbol{u}_{p+1} \perp \boldsymbol{u}_{p+2}$.

That is,

$$\begin{aligned}
\hat{\boldsymbol{u}}_{p+1} =& \boldsymbol{g} - \sum_{i=1}^{p} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{g}) \\
\boldsymbol{u}_{p+1} =& \hat{\boldsymbol{u}}_{p+1}/\|\hat{\boldsymbol{u}}_{p+1}\|, \quad \boldsymbol{u}_{p+1}(\boldsymbol{u}_{p+1}^T \boldsymbol{g}) = \boldsymbol{g} - \sum_{i=1}^{p} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{g}) \\
\hat{\boldsymbol{u}}_{p+2} =& \boldsymbol{h} - \sum_{i=1}^{p+1} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{h}), \quad \boldsymbol{u}_{p+2} = \hat{\boldsymbol{u}}_{p+2}/\|\hat{\boldsymbol{u}}_{p+2}\|.
\end{aligned} \tag{3.18}$$

Then ξ_{22} and η_2 can be simplified as

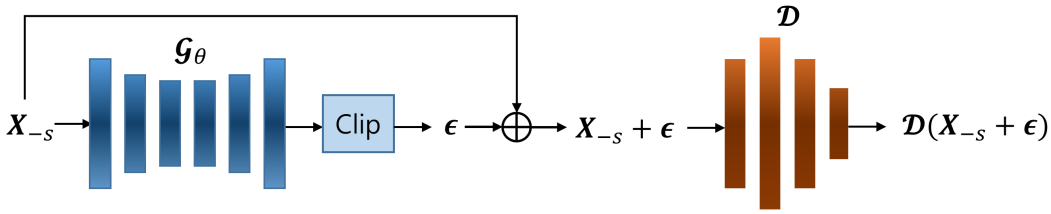


Figure 3.2: Architecture for adversarial perturbation against attribute inference attack

$$
\begin{aligned}
\xi_{22} =& \sum_{j=1}^{n} \frac{(\boldsymbol{g}_{s2}^T \boldsymbol{u}_j)^2}{\sigma_j^2 + \lambda} = \sum_{j=1}^{p} \frac{(\boldsymbol{g}_{s2}^T \boldsymbol{u}_j)^2}{\sigma_j^2 + \lambda} + \frac{1}{\lambda} \boldsymbol{g}_{s2}^T \boldsymbol{u}_{p+1} (\boldsymbol{u}_{p+1}^T \boldsymbol{g}_{s2}) \\
=& \frac{1}{\lambda} \left[- \sum_{j=1}^{p} \frac{\sigma_j^2 (\boldsymbol{g}_{s2}^T \boldsymbol{u}_j)^2}{\sigma_j^2 + \lambda} + \boldsymbol{g}_{s2}^T \boldsymbol{g}_{s2} \right] \\
\eta_2 =& \sum_{j=1}^{n} \frac{(\boldsymbol{g}_{s2}^T \boldsymbol{u}_j)(\boldsymbol{u}_j^T \boldsymbol{y})}{\sigma_j^2 + \lambda} = \sum_{j=1}^{p} \frac{(\boldsymbol{y}^T \boldsymbol{u}_j)(\boldsymbol{u}_j^T \boldsymbol{g}_{s2})}{\sigma_j^2 + \lambda} + \frac{1}{\lambda} \boldsymbol{y}^T (\boldsymbol{g}_{s2} - \sum_{j=1}^{p} \boldsymbol{u}_j (\boldsymbol{u}_j^T \boldsymbol{g}_{s2})) \\
=& \frac{1}{\lambda} \left[- \sum_{j=1}^{p} \frac{\sigma_j^2 (\boldsymbol{g}_{s2}^T \boldsymbol{u}_j)(\boldsymbol{y}^T \boldsymbol{u}_j)}{\sigma_j^2 + \lambda} + \boldsymbol{y}^T \boldsymbol{g}_{s2} \right].
\end{aligned} \tag{3.19}
$$

Therefore the ridge estimate can be further simplified as

$$
\begin{aligned}
\hat{\boldsymbol{f}} =& \sum_{j=1}^{p} \frac{\sigma_j^2}{\sigma_j^2 + \lambda} \boldsymbol{u}_j (\boldsymbol{u}_j^T \boldsymbol{y}) + \frac{(\eta_1 + \xi_{22}\eta_1 - \xi_{12}\eta_2)}{(1 + \xi_{11})(1 + \xi_{22}) - \xi_{12}^2} (\boldsymbol{h}_{s1} - \sum_{j=1}^{p} \frac{\sigma^2}{\sigma_j^2 + \lambda} \boldsymbol{u}_j (\boldsymbol{u}_j^T \boldsymbol{h}_{s1})) \\
&+ \frac{(\eta_2 + \xi_{11}\eta_2 - \xi_{12}\eta_1)}{(1 + \xi_{11})(1 + \xi_{22}) - \xi_{12}^2} (\boldsymbol{g}_{s2} - \sum_{j=1}^{p} \frac{\sigma^2}{\sigma_j^2 + \lambda} \boldsymbol{u}_j (\boldsymbol{u}_j^T \boldsymbol{g}_{s2})).
\end{aligned} \tag{3.20}
$$

3.3.3 Adversarial Perturbation Against Attribute Inference Attack

Let a classification network $\mathcal{A}$ denote the attacker's classifier. For one sensitive variable $\boldsymbol{x}_s$ and non-sensitive variables $\boldsymbol{X}_{ns}$, let public and private datasets $D^{pub} = (\boldsymbol{X}_{ns}^{pub}, \boldsymbol{x}_s^{pub})$ and $D^{priv} = (\boldsymbol{X}_{ns}^{priv}, \boldsymbol{x}_s^{priv})$. $\mathcal{A}$ is trained to minimize a loss function $\mathcal{L}_{att}(\mathcal{A}(\boldsymbol{X}_{ns}^{pub}), \boldsymbol{x}_s^{pub})$. $\mathcal{L}_{att}$ can be determined depending on what kind of variable $\boldsymbol{x}_s$ is. For example, the attacker can use cross entropy loss if $\boldsymbol{x}_s$ is a discrete variable, and the MSE loss can be used if it is a continuous variable. For t sensitive variables, the attacker can train t models separately.

On the other hand, the goal of the defender is to maximize $\mathcal{L}_{att}(\mathcal{A}(\boldsymbol{X}_{ns}^{priv} + \boldsymbol{\epsilon}), \boldsymbol{x}_s^{priv})$ by adding a little noise $\boldsymbol{\epsilon}$ to $\boldsymbol{X}_{ns}^{priv}$. Because the defender does not have information of the model the attacker used, the defender trains a model $\mathcal{D}$ with D^{pub} and use it as an alternative of $\mathcal{A}$. As explained in [91], using a surrogate model is plausible because of the transferability between the two models. Based on the evasion attack which is a kind of an adversarial attack, [91] considered $\boldsymbol{\epsilon}$ to be the minimum noise that changes the classification result of $\boldsymbol{X}_{ns}^{priv}$. However, their method has a fundamental problem that it cannot be applied for a continuous $\boldsymbol{x}_s$. Also, it has a problem of scalability because an optimization should be solved for each row of $\boldsymbol{X}_{ns}^{priv}$.

To address both of the above problems, we propose a noise generating network $\mathcal{G}_\theta$ as

$$\mathcal{G}_\theta(\boldsymbol{X}_{ns}) = \mathbb{E}[\boldsymbol{\epsilon}|\boldsymbol{X}_{ns}]. \tag{3.21}$$

Because the dimensionality of the noise is same as the dimensionality of the input, we use an autoencoder structure for $\mathcal{G}_\theta$. The generator $\mathcal{G}_\theta$ is trained to maximize

the loss of the defender $\mathcal{D}$, i.e.,

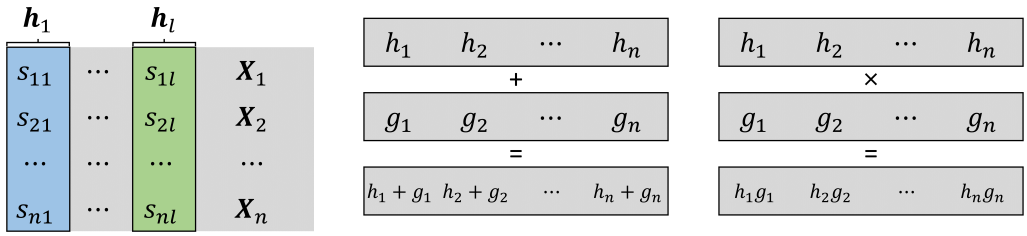
$$\theta = \arg\max_{\theta'} \mathcal{L}_{def}(\mathcal{D}(\boldsymbol{X}_{ns}^{priv} + \mathcal{G}_{\theta}(\boldsymbol{X}_{ns}^{priv})), \boldsymbol{x}_{s}^{priv}). \tag{3.22}$$

We note that it is also possible to make the output of $\mathcal{G}_\theta$ become (input+noise). However, in this case it is difficult to directly limit the magnitude of the noise. On the other hand, if the output of $\mathcal{G}_\theta$ is used as noise as in (3.22), the noise level can be directly adjusted by clipping the output to the desired magnitude. The overall flow of our defense method is depicted in Fig 3.2.

Our defense method can be seen as adopting a generative adversarial perturbation (GAP) [143], a kind of adversarial attack which generates an adversarial attack with a neural network, to the task of defending against attribute inference attacks. However, our method can support both classification model for discrete sensitive variables and regression model for continuous sensitive variables, whereas GAP concentrates on attacking a classification model.

3.3.4 Algorithm for Ridge Regression

Because (3.20) requires as many summations as the number of variables, we put each sensitive variable in a different ciphertext, and all values for each column are grouped into a single ciphertext. Some details about our ciphertext packing is presented in the appendix. The maximum number of plaintext values that can be packed into one ciphertext is determined based on the security parameter. Here, we assume that the length of the columns does not exceed this parameter. Otherwise, each column should be divided into multiple ciphertexts, but this does not significantly affect the complexity of the algorithm. When a ciphertext contains a scalar value such as η_1



(a) Packing (b) Addition (c) Multiplication

Figure 3.3: Description of the ciphertext packing and operations between ciphertexts. The values in the thick box are packed into the same ciphertext.

or ξ_{11}, it can be thought of as a vector that repeats the same value as many times as the number of slots. Our packing strategy, and addition and multiplication between ciphertexts are depicted in Fig 3.3.

Algorithm 1 demonstrates the use of our method for a ridge regression with two encrypted sensitive variables. The basic operations used in the algorithm are presented in the appendix. For convenience, Rescale after any multiplication is omitted. Using Rotate, the sum of n elements in a ciphertext can be calculated within $\log n$ times. Therefore, the complexity of Algorithm 1 is $O(d \log n)$. Inv is an evaluation of Goldschmidt's division algorithm with τ times of iterations.

3.3.5 Algorithm for Adversarial Perturbation

Algorithm 2 presents the procedures for defending against attribute inference attacks. The first phase of the algorithm is to train an attack model which aims to predict the value of sensitive variables. With the trained attack model, in the second phase the noise generator network is trained to degrade the performance of the attack model for private data. We note that Cl in the second phase is the clipping function which bounds the norm of the generated noise to C. Any norm such as L_1-

Algorithm 1 Ridge Regression for two Sensitive Variables

procedure RidgeEstimate($\mathbf{h}_1, \mathbf{h}_2, \{\mathbf{u}_i\}, \{\sigma_i\}, \mathbf{y}, \lambda, \tau$) ▷ calculate the ridge estimate
  for s in $1, 2$ **do**
    for i in $1, \ldots, d$ **do**
      $tmp \leftarrow \mathsf{ConstMult}(\mathbf{h}_s, \mathbf{u}_i)$
      for j in $1, \ldots, \log_2 n$ **do**
        $tmp\_rot \leftarrow \mathsf{Rotate}(tmp, -2^j)$
        $tmp \leftarrow \mathsf{Add}(tmp, tmp\_rot)$
      end for
      $tmp \leftarrow \mathsf{ConstMult}(tmp, \frac{\sigma_i^2}{\sigma_i^2+\lambda}\mathbf{u}_i)$
      if i=0 **then**
        $\mathbf{c}_s \leftarrow tmp$
      else
        $\mathbf{c}_s \leftarrow \mathsf{Add}(\mathbf{c}_s, tmp)$
      end if
    end for
    $\mathbf{c}_s \leftarrow \mathsf{Sub}(\mathbf{h}_s, \mathbf{c}_s)$
  end for
  $c_3 \leftarrow \mathsf{ConstMult}(\mathbf{c}_1, \frac{1}{\lambda}\mathbf{y})$; $c_4 \leftarrow \mathsf{ConstMult}(\mathbf{c}_2, \frac{1}{\lambda}\mathbf{y})$
  $c_5 \leftarrow \mathsf{Mult}(\mathbf{c}_1, \mathsf{ConstMult}(\mathbf{h}_1, \frac{1}{\lambda}))$; $c_6 \leftarrow \mathsf{Mult}(\mathbf{c}_2, \mathsf{ConstMult}(\mathbf{h}_2, \frac{1}{\lambda}))$;
  $c_7 \leftarrow \mathsf{Mult}(\mathbf{c}_1, \mathsf{ConstMult}(\mathbf{h}_2, \frac{1}{\lambda}))$
  for i in $3, \ldots, 7$ **do**
    for j in $1, \ldots, log_2 n$ **do**
      $c_i\_rot \leftarrow \mathsf{Rotate}(c_i, -2^j)$
      $c_i \leftarrow \mathsf{Add}(c_i, c_i\_rot)$
    end for
  end for
  $c_8 \leftarrow \mathsf{Sub}(\mathsf{Add}(c_3, \mathsf{Mult}(c_3, c_7)), \mathsf{Mult}(c_4, c_6))$
  $c_9 \leftarrow \mathsf{Sub}(\mathsf{Add}(c_4, \mathsf{Mult}(c_4, c_5)), \mathsf{Mult}(c_3, c_6))$
  $c_{10} \leftarrow \mathsf{Inv}(\mathsf{Sub}(\mathsf{Mult}(\mathsf{ConstAdd}(\mathsf{c_5}, 1), \mathsf{ConstAdd}(c_7, 1)), \mathsf{Mult}(c_6, c_6)), \tau)$
  $\hat{\boldsymbol{r}} \leftarrow \mathsf{Add}(\mathsf{Mult}(c_{10}, \mathsf{Mult}(c_8, c_1)), \mathsf{Mult}(c_{10}, \mathsf{Mult}(c_9, c_2)))$
  $\hat{\boldsymbol{r}} \leftarrow \mathsf{ConstAdd}(\hat{\boldsymbol{r}}, \sum_{i=1}^{d} \frac{\sigma_j^2}{\sigma_j^2+\lambda}\boldsymbol{u}_j\boldsymbol{u}_j^T\boldsymbol{y})$
  return $\hat{\boldsymbol{r}}$
end procedure

norm and L_2-norm can be used for clipping, and the trade-off between the degree of defense and the regression utility can be controlled by adjusting the value of C.

Algorithm 2 Defense against Attribute Inference Attack

procedure TrainAttack(public non-private data $\{X_{ns}^{pub}\}_{k=1}^{n_{pub}}$, public sensitive variable $\{x_s^{pub}\}_{k=1}^{n_{pub}}$, number of epochs e_1, batch size b_1, learning rate ℓ_1)
    Initialize parameters ϕ of $\mathcal{D}_\phi$
    for $1, \ldots, e_1$ **do**
        for $1, \ldots, \lceil n_{pub}/b_1 \rceil$ **do**
            Sample mini-batches of b_1 samples B
            $\mathcal{L} = \sum_{(X_{ns}^{pub}, x_s^{pub}) \in B} \mathcal{L}_{def}(\mathcal{D}_\phi(X_{ns}^{pub}), x_s^{pub})$
            $\phi \leftarrow \phi - \ell_1 \nabla_\phi \mathcal{L}$
        end for
    end for
end procedure
procedure TrainNoise(private non-private data $\{X_{ns}^{priv}\}_{k=1}^{n_{priv}}$, private sensitive variable $\{x_s^{priv}\}_{k=1}^{n_{priv}}$, number of epochs e_2, batch size b_2, learning rate ℓ_2, clipping parameter C)
    Initialize parameters θ of $\mathcal{G}_\theta$
    for $1, \ldots, e_2$ **do**
        for $1, \ldots, \lceil n_{priv}/b_2 \rceil$ **do**
            Sample mini-batches of b_2 samples B
            $\mathcal{L} = \sum_{(X_{ns}^{priv}, x_s^{priv}) \in B} \mathcal{L}_{def}(\mathcal{D}_\phi(X_{ns}^{priv} + \mathsf{cl}(\mathcal{G}_\theta(X_{ns}^{priv}), C)), x_s^{priv})$
            $\theta \leftarrow \theta + \ell_2 \nabla_\theta \mathcal{L}$
        end for
    end for
end procedure

3.4 Experiments

3.4.1 Experimental Setting

Datasets We used ten real-world datasets from the UCI machine learning repository [49], along with another dataset from [2]. Among the 11 datasets, 7 were originally generated for classification tasks and have binary target variables. Before encryption, all input variables, including sensitive variables, were scaled to have values within $[0, 1]$. In addition, we used only numerical and binary categorical variables. We considered this to be more advantageous to the compared method, because the storage cost of our approach does not significantly decrease even if the number of variables decreases, due to the fact that our method only encrypts a fixed number of sensitive variables. A detailed description of the datasets is shown in Table 3.1.

Because the criterion for determining a sensitive variable differs from study to study, we measured the feature importance through a ridge regression on plaintext with Python and treated the variable with higher feature importance as a sensitive variable. In real situations, the data owner can determine the sensitive variables according to their level of importance.

Compared methods We compared our method with three methods as follows:

- **Baseline**: A ridge regression method without any encrypted variable. It serves as a benchmark for regression performance.
- **FullColumn**: A method that encrypts all input variables implemented with CKKS. Following [128, 8], it was trained using a gradient descent.
- **Ours-1**: Our proposed ridge regression method with one encrypted sensitive

Table 3.1: Description of the datasets

Dataset	Columns	Total Data points	Binary Target variable	Learning rate (FullColumn)
Australian Credit (**Credit**)	10	690	yes	0.01
Bupa Liver Disorders (**Disorders**)	5	345	no	0.1
Heart Disease (**Heart**)	8	303	yes	0.01
Pima Indians Diabetes (**Indians**)	8	768	yes	0.01
Wisconsin Breast Cancer (**Cancer**)	9	698	yes	0.001
Bank Marketing (**Bank**)	10	4521	yes	0.001
Student Performance (**Student**)	24	395	no	0.01
Graduate Admission (**Admission**)	7	400	no	0.01
Diabetes (**Diabetes**)	10	442	no	0.01
Adult (**Adult**)	11	30162	yes	0.0001
Census Income (**Census**)	18	199523	yes	0.00001

variable.

- **Ours-2**: Our proposed ridge regression method with two encrypted sensitive variables.

Experimental details All experiments were carried on a machine using 40 threads of an Intel Xeon E-2660 v3 @2.60_GHz CPU processors. For the CKKS parameters, we set $\log q_L = 1200$, $\log p = 40$ and $N = 2^{16}$. We tested our method by changing the value of λ in $\{0.1, 0.5, 1, 5, 10\}$. The learning rates for the gradient descent method were predetermined within the range of $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ before the experiment with plaintext to show the best performance, and the number of iterations was fixed to 8 to avoid bootstrapping. The selected learning rates for each dataset are shown in Table 3.1.

Because the efficiency of homomorphic encryption is highly dependent on the packing strategy for SIMD operations, for a fair comparison we packed each column (attribute) into one ciphertext for all datasets except **Census**. Because the number of training data is larger than the maximum number of plaintext values that can be

packed into one ciphertext ($= N/2$), for **Census** dataset we split each column into three ciphertexts. We empirically compared the computational time and R^2 score of our method with those of **1Sens** and **FullColumn**. All experiments were repeated five times, and the average of each measurement was recorded. We split each dataset into three sets. First, 50% of each dataset was randomly sampled as the public set. The public set was used for training classifiers of the attacker and the defender, and the defense method. Then, we used 80% of the remaining data as the training set for the ridge regression, and the remaining 20% as the test set. The models of the attacker and the defender were set as feed-forward neural networks with one hidden layer, and the defense method was set as an autoencoder with one hidden layer. The size of the hidden layer of the attacker's network was set to be 3 times the number of the variables, and the size of the hidden layer of the other networks were set to be 5 times the number of the variables.

3.4.2 Experimental Results

Table 3.2 shows the computation time for the 11 datasets with a regularization parameter $\lambda = 1$. It is shown that owing to encrypting only sensitive variables, our method is 5-20 times faster than **FullColumn** depending no the number of variables. In addition to the performance of each method, we plotted in Fig 3.4 the performance ratio of **FullColumn** and **Ours-2**, as well as the performance ratio of **Ours-2** and **Ours-1**.

Because the computation cost of each method is roughly proportional to the number of columns, and because we packed each column as a ciphertext, we predicted that the performance ratio would be proportional to the number of columns in the

dataset. Therefore, if the ratio value is higher than the number of variables divided by 2, **Ours-2** is more efficient, and if the value is lower, the opposite can be considered. Fig 3.4(a) demonstrates that **Ours-2** was always more efficient than **FullColumn**, especially with fewer variables. In addition, our method does not require searching from the learning rate because it does not use a gradient descent. Although is difficult to quantitatively express the time taken for parameter search, we emphasize that this fact makes our method far more efficient.

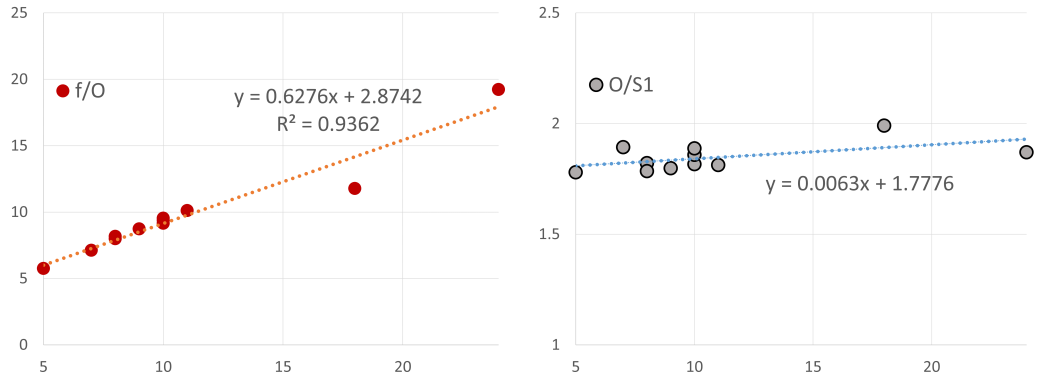
Fig 3.4(b) shows that **Ours-2** required about 1.7-1.9 times of computation time compared to **Ours-1**. The reason that ratio is always smaller than 2 is that **Ours-2** exploits one less non-sensitive variable by assuming two sensitive variables. Because Algorithm 1 requires as many iterations as the number of non-sensitive variables, the computation time of **Ours-2** is slightly reduced. It can be seen from the figure that the ratio tends to approach 2 as the number of variables increases. The results for different values of λ are provided in Table 3.3 to Table 3.6.

Meanwhile, the results for **Census** dataset in which each column is divided into multiple ciphertexts deviate slightly from the overall trend (second point from the right in Fig 3.4). If the number of ciphertexts for each column increases, the operation time of all methods increases approximately in proportion to it. However, not every part of the algorithm becomes slower. Taking Algorithm 1 as an example, while the number of operations to calculate c_1 to c_7 (line 2 to line 26) increases by almost exactly the number of ciphertexts for each column, the next part is no longer affected by the number of ciphertexts. Because the part that is affected by the number of ciphertexts is relatively larger than the compared methods, **Ours-2** becomes less efficient when the number of data is very large, as shown in Fig 3.4.

The R^2 score is a metric widely used to measure the performance of regression models. The higher the value is, the higher the prediction performance is and it takes a value of 1 when the model achieves a perfect prediction. The R^2 score of each model ($\lambda = 1$) is also shown in Table 3.2. The table also contains the regression results for **Baseline**. Except for three datasets, **Ours** consistently performed better than **FullColumn**. Even when **FullColumn** had a higher R^2 score, the R^2 score of **Ours** was more similar to that of **Baseline**. The results show that using a gradient descent, it is difficult to converge to the exact solution using only a small number of iterations. Although in some cases, by chance, the approximate solution performs better than the actual solution, in most cases it does not. The difference between the R^2 score of **Baseline** and **Ours** is mainly due to the small number of iterations in Inv, and partly due to the approximation of CKKS scheme. The results for different values of λ are provided in Table 3.3 to Table 3.6.

We additionally performed t-test between **FullColumn** and the proposed method for $\lambda = 1$ and recorded the results in Table 3.7. Except for **Heart**, **Cancer**, and **Student** datasets where **FullColumn** showed better regresson results than the proposed method, it was shown that both **Ours-1** and **Ours-2** were significantly better ($p < 0.001$) than **FullColumn**.

Defense against attribute inference attacks In this section we evaluate our adversarial perturbation method (**AP**) with one sensitive variable. We compared **AP** with **Attriguard** proposed in [91]. Because **Attriguard** can only be applied to discrete sensitive variables, the comparison was made only for four datasets, **Credit**, **Heart**, **Adult**, and **Census**. We considered the cross entropy with the



(a) Comparison with **Fullcolumn**

(b) Comparison with **Ours-1**

Figure 3.4: Visualization of the results for the computation time. The x-axis indicates the number of variables, and the y-axis indicates the ratio of the computation times.

target variable, which is mainly used for classification task, as the loss function $\mathcal{L}_{def}$ (**AP-CE**), but following **Attriguard**, the cross entropy with the prior distribution of the sensitive variable was also considered as a loss function (**AP-KL**).

The first column of Fig 3.5 plots the attacker's inference accuracy for the noise budget C. For **Credit** dataset, all three methods lowered the attacker's accuracy by similar values. For very low C values, **Attriguard** showed the best performance while **AP-KL** showed bad performance. For **Heart** dataset, only **AP-CE** defended well against the attack. For **Adult** dataset, the attacker's accuracy converged for all methods, but **Attriguard** recorded the lowest convergence value. For **Census** dataset, **AP-CE** was always better than **Attriguard**, and **AP-KL** showed the best defense at $C = 2.0$.

Although the utility loss due to the noise can be measured as the magnitude of the noise, it may be more desirable to measure how much it affects the actual performance of ridge regression. The second column of Fig 3.5 plots the R^2 score

Table 3.2: Results for computation time and regression performance with $\lambda = 1$.

Dataset	Time (sec)			R^2 score			
	FullColumn	**Ours-1**	**Ours-2**	**FullColumn**	**Ours-1**	**Ours-2**	**Baseline**
Credit	744.466	42.951	78.029	0.5162	**0.5178**	**0.5178**	0.5177
Disorders	245.907	24.039	42.789	0.1416	**0.1573**	**0.1573**	0.1573
Heart	452.905	31.141	56.707	**0.1974**	0.1739	0.1737	0.1740
Indians	533.078	36.556	65.258	0.2056	**0.2421**	**0.2421**	0.2421
Cancer	630.344	40.146	72.184	**0.7729**	0.7644	0.7492	0.7644
Bank	851.390	50.069	93.076	-0.0283	**0.1757**	**0.1757**	0.1757
Student	2959.940	82.322	153.976	**0.0810**	0.0676	0.0660	0.0677
Admission	396.904	29.428	55.701	0.7635	0.8118	**0.8136**	0.8137
Diabetes	684.247	38.670	73.004	0.3972	**0.4308**	**0.4308**	0.4308
Adult	1166.396	63.678	115.406	0.2359	**0.3230**	**0.3230**	0.3230
Census	7983.130	343.381	680.125	0.1036	**0.1948**	0.1947	0.1948

of ridge regression for C. For all datasets, it was found that **AP-CE** maintained the R^2 score better than **Attriguard**. In particular, for **Credit** dataset, although **Attriguard** defended against the attack with a smaller noise, the performance of ridge regression was rather lower. The result seems to be because **AP-CE** searches a larger probability space for the noise than **Attriguard**, which finds only one noise per each value of sensitive variable.

For continuous sensitive variable, we used MSE as the loss function. The results for those variables are provided in Figure 3.6. It is commonly shown that for all the datasets **AP** effectively increased the attacker's loss by a moderate sacrifice of the performance of ridge regression.

Table 3.3: Results for computation time and regression performance with $\lambda = 0.1$.

Dataset	Time (sec)			R^2 score			
	FullColumn	**Ours-1**	**Ours-2**	**FullColumn**	**Ours-1**	**Ours-2**	**Baseline**
Credit	757.938	42.958	78.024	0.5159	**0.5170**	0.5169	0.5146
Disorders	242.810	23.502	41.849	0.1616	0.2170	**0.2181**	0.2192
Heart	454.040	31.159	57.386	**0.1998**	0.1506	0.1494	0.1506
Indians	529.409	36.753	63.641	0.2093	**0.2330**	0.2251	0.2330
Cancer	629.056	40.116	72.194	**0.7733**	0.7609	0.7609	0.7609
Bank	847.323	50.953	93.129	-0.0284	**0.1761**	**0.1761**	0.1761
Student	2929.000	83.062	154.588	**0.0792**	0.0449	0.0462	0.0462
Admission	395.290	29.510	56.615	0.7658	**0.8203**	**0.8203**	0.8203
Diabetes	684.568	38.049	72.691	0.4025	**0.4217**	0.4216	0.4217
Adult	1165.470	61.642	114.676	0.2359	**0.3233**	0.3232	0.3232
Census	7913.180	343.083	649.067	0.1036	**0.1948**	**0.1948**	0.1948

Table 3.4: Results for computation time and regression performance with $\lambda = 0.5$.

Dataset	Time (sec)			R^2 score			
	FullColumn	**Ours-1**	**Ours-2**	**FullColumn**	**Ours-1**	**Ours-2**	**Baseline**
Credit	743.300	43.743	79.265	0.5161	0.5099	**0.5192**	0.5163
Disorders	249.023	23.462	43.747	0.1522	**0.1847**	0.1844	0.1847
Heart	452.676	31.259	56.785	**0.1988**	0.1632	0.1614	0.1632
Indians	534.419	37.004	66.870	0.2077	**0.2332**	0.2318	0.2389
Cancer	634.045	40.317	73.069	**0.7732**	0.7626	0.7626	0.7626
Bank	862.840	50.174	94.006	-0.0283	**0.1759**	**0.1759**	0.1759
Student	2961.280	82.137	155.392	**0.0800**	0.0575	0.0575	0.0575
Admission	393.240	29.337	55.781	0.7648	0.8179	**0.8180**	0.8180
Diabetes	678.350	39.419	72.903	0.4001	0.4202	**0.4273**	0.4273
Adult	1164.760	62.536	113.801	0.2359	0.3232	**0.3246**	0.3231
Census	7936.260	336.299	685.656	0.1036	**0.1948**	0.1943	0.1948

3.5 Chapter Summary

We proposed a privacy-preserving ridge regression algorithm with homomorphic encryption of multiple private variables. In addition, we suggested an adversarial perturbation method which can defend attribute inference attacks on the private variables. The experimental results showed that the computational time of our algorithm is faster in proportion to the number of variables than the gradient descent method. Moreover, compared to an existing defense method, our adversarial perturbation

Table 3.5: Results for computation time and regression performance with $\lambda = 5$.

Dataset	Time (sec)			R^2 score			
	FullColumn	**Ours-1**	**Ours-2**	**FullColumn**	**Ours-1**	**Ours-2**	**Baseline**
Credit	740.849	42.767	78.723	0.5161	**0.5191**	**0.5191**	0.5191
Disorders	246.376	23.943	42.801	**0.0918**	**0.0918**	0.0891	0.0918
Heart	451.372	30.740	56.695	0.1842	**0.1904**	**0.1904**	0.1904
Indians	542.671	35.884	67.632	0.1897	0.2220	**0.2221**	0.2221
Cancer	633.365	40.231	71.659	0.7681	**0.7716**	0.7692	0.7716
Bank	844.614	49.746	91.668	-0.0278	**0.1739**	0.1738	0.1739
Student	2965.350	81.926	152.499	0.0860	0.0977	**0.0981**	0.0981
Admission	401.839	29.751	53.882	0.7504	**0.7782**	**0.7782**	0.7782
Diabetes	698.430	39.075	74.380	0.3721	0.3849	**0.4070**	0.4096
Adult	1164.140	65.187	114.903	0.2358	0.3224	**0.3234**	0.3223
Census	8035.450	342.837	680.125	0.1037	**0.1946**	**0.1946**	0.1946

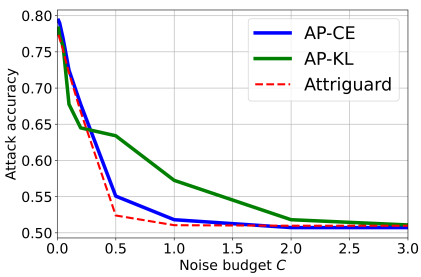
Table 3.6: Results for computation time and regression performance with $\lambda = 10$.

Dataset	Time (sec)			R^2 score			
	FullColumn	**Ours-1**	**Ours-2**	**FullColumn**	**Ours-1**	**Ours-2**	**Baseline**
Credit	740.320	42.745	78.529	0.5127	**0.5146**	**0.5146**	0.5146
Disorders	245.705	25.111	42.637	0.0643	**0.0655**	**0.0655**	0.0655
Heart	450.651	31.255	56.102	0.1641	**0.1713**	0.1711	0.1713
Indians	528.869	36.966	63.694	0.1713	0.1884	**0.1891**	0.1891
Cancer	632.784	40.219	73.061	0.7532	**0.7730**	0.7723	0.7731
Bank	859.006	49.864	93.128	-0.0272	**0.1690**	**0.1690**	0.1690
Student	2941.460	81.514	153.395	0.0874	0.0951	**0.0989**	0.0989
Admission	397.849	29.515	56.652	0.7295	**0.7433**	0.7430	0.7433
Diabetes	681.683	38.734	71.069	0.3398	**0.3635**	0.3633	0.3635
Adult	1171.130	64.754	119.145	0.2358	**0.3214**	**0.3214**	0.3214
Census	7849.710	336.474	688.455	0.1037	**0.1938**	**0.1938**	0.1938

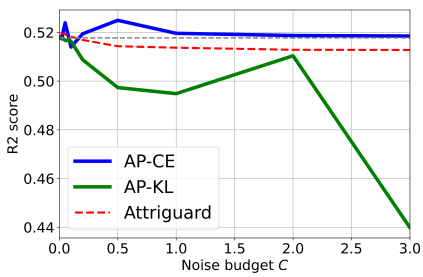
method can effectively prevent inference attacks while preserving the performance of the ridge regression. Our method can be developed more efficiently by using other plaintext packing strategies, or by designing more sophisticated SIMD operations. The extention of the proposed method to other machine learning algorithms needs to be further investigated.

Table 3.7: T-test results between FullColumn and Ours with $\lambda = 1$.

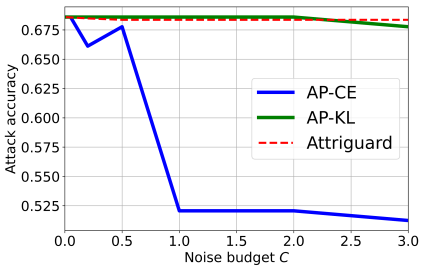
Dataset	Ours-1 >FullColumn		Ours-2 >FullColumn	
	t-statistics	p value	t-statistics	p value
Credit	1.694	0.061	-0.062	0.524
Disorders	7.148	<0.001	8.475	<0.001
Heart	-11.028	0.999	-12.278	0.999
Indians	17.607	<0.001	17.274	<0.001
Cancer	-7.063	0.999	-24.292	0.999
Bank	170.994	<0.001	132.112	<0.001
Student	-8.597	0.999	-8.791	0.999
Admission	41.087	<0.001	33.637	<0.001
Diabetes	21.400	<0.001	22.464	<0.001
Adult	58.309	<0.001	69.593	<0.001
Census	69.525	<0.001	58.416	<0.001



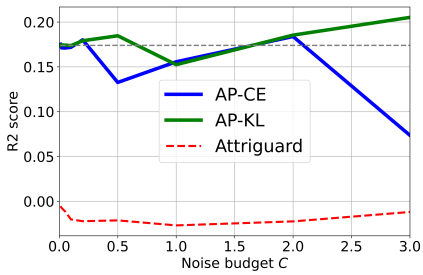
(a) **Credit-Attack**



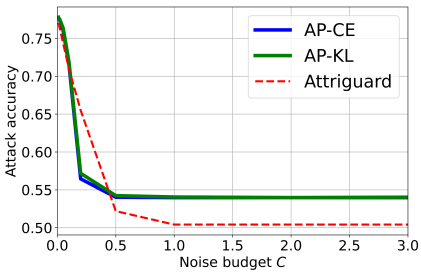
(b) **Credit-Regression**



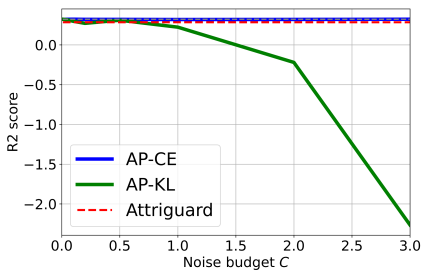
(c) **Heart-Attack**



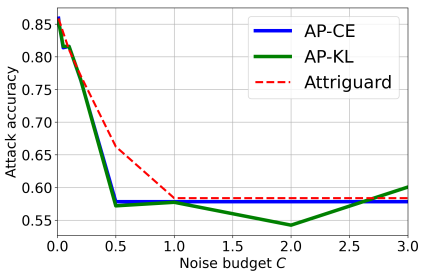
(d) **Heart-Regression**



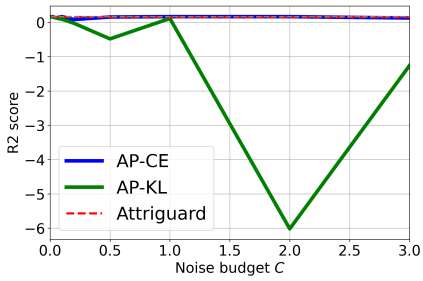
(e) **Adult-Attack**



(f) **Adult-Regression**

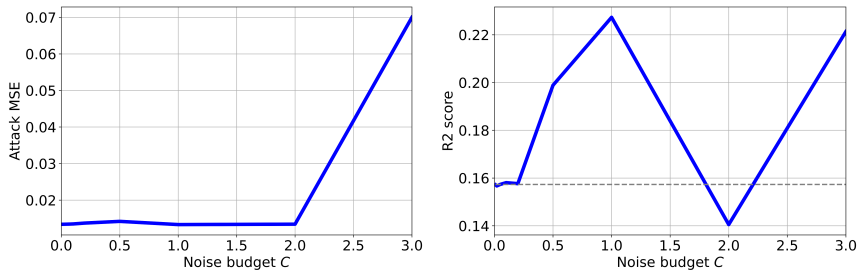


(g) **Census-Attack**

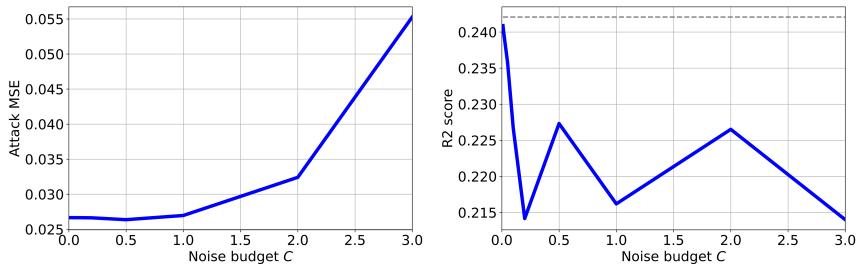


(h) **Census-Regression**

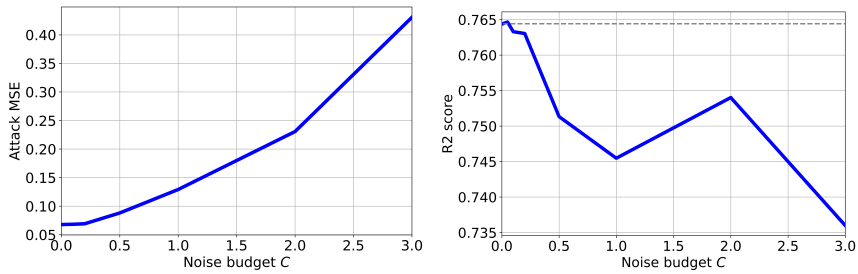
Figure 3.5: Defense results against attribute inference attack (first column) and ridge regression results with defense methods (second column) for datasets with discrete sensitive variable.



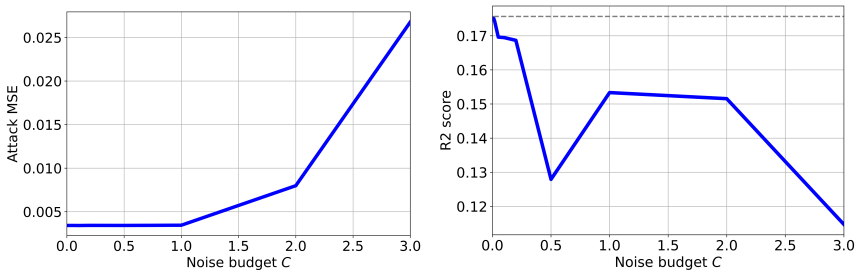
(a) **Disorders**



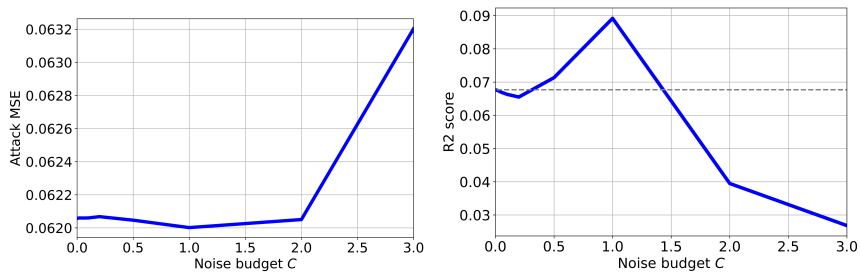
(b) **Indians**



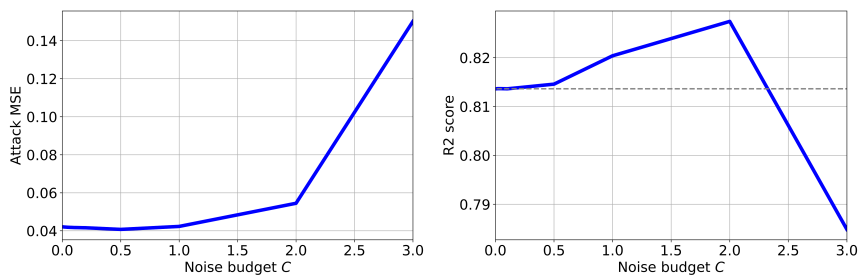
(c) **Cancer**



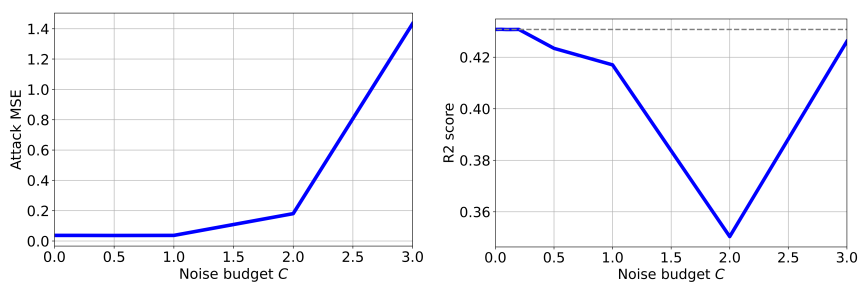
(d) **Bank**



(e) **Student**



(f) **Admission**



(g) **Diabetes**

Figure 3.6: Defense results against attribute inference attack (first column) and ridge regression results with defense methods (second column) for datasets with continuous sensitive variable.

Chapter 4

Parameter-free Homomorphic-encryption-friendly Logistic Regression

4.1 Problem Statement

Machine learning on encrypted data (MLED) is an effective method to ensure privacy for both machine learning models and data used for the model training. Unlike other privacy-preserving machine learning methods, MLED does not require decryption of intermediate products of the algorithm or data transfer between participants, enabling a complete outsourcing of machine learning. As data have become increasingly valuable, MLED models have been actively studied in an attempt to combine data and computing power that are separated from each other. With the development of efficient homomorphic encryption (HE) schemes that enable MLED, recent studies [30, 42, 146, 119, 118, 121] cover most machine learning models, from simple linear regression to deep neural networks (DNN).

However, current research on MLEDs remains lacking owing to the limitations of the proposed HE schemes. Operations on ciphertexts are very inefficient in terms of computation time and memory consumption, compared to operations on plaintexts. In addition, after performing a certain number of operations, a costly procedure called bootstrapping is required [76, 31]. Therefore, the majority of studies are lim-

ited to the inference phase, assuming a situation that provides machine learning inference as a service. The training phase should be studied more deeply top extract the maximum value from the data.

One of the MLED models, whose training steps have been commonly studied, is logistic regression (LR). LR is a simple classification model that linearly divides the data space, which is used effectively in various fields, including medicine, marketing, and geology [40, 47, 116]. [104] trained an encrypted LR model with gradient descent and approximated the sigmoid function with the most similar polynomials within a certain range. Despite performing well on several benchmark datasets, their methodology raises some concerns because the training parameters, such as learning rate, number of iterations, and range of sigmoid function approximation should be predetermined. Indeed, this problem is common for all proposed MLED training models.

In this study, we propose an effective privacy-preserving LR method that is free from most hyperparameter selections. First, we train an unencrypted classification model to extract the prediction probability for each label. Then, we solve for a ridge regression that predicts the logit result of the estimated probability. Our theoretical results provide a new generalization error bound, which can be optimized by properly estimating the logit and our proposed mean matching, which aims to reduce the gap between the distributions of encrypted and unencrypted data. To benefit from the trade-off between security and efficiency, we define sensitive variables, whose privacy is more important than the privacy of others, which has been widely recognized in other fields but has rarely been introduced into MLED. The experimental results show that, compared to the prior encrypted LR models, our method achieves better

classification results and lower training latency.

4.2 Proposed Method

4.2.1 Motivation

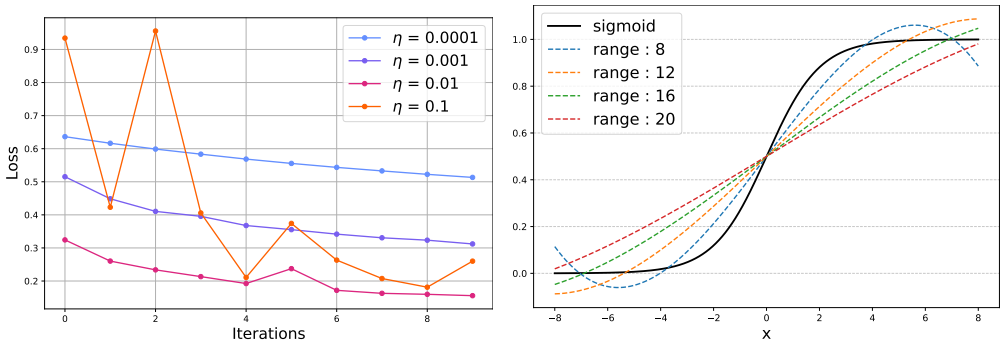
The loss function of LR is the negative log-likelihood, which is given by

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log(1 + \exp(-y^i \boldsymbol{\theta}^T \boldsymbol{x}^i)) \tag{4.1}$$

where $\boldsymbol{\theta}$ is the weight vector, $\boldsymbol{x}^i$ is i-th input and $y^i \in \{+1, -1\}$ is the label of i-th input. [101] minimized (4.1) with Nesterov's accelerated gradient, which is a slight modification of the gradient descent algorithm. The gradient descent algorithm can be written as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{g}_t, \quad \boldsymbol{g}_t = \frac{\partial J}{\partial \boldsymbol{\theta}} = -\sum_{i=1}^{n} \text{sig}(-y^i \boldsymbol{\theta}_t^T \boldsymbol{x}^i) \cdot y^i \boldsymbol{x}^i \tag{4.2}$$

where η is the learning rate and $\text{sig}(x) = 1/(1 + \exp(-x))$ is the sigmoid function. When implementing this model with HE, there are two major limitations. The first is the selection of the learning rate and number of iterations for the gradient descent. In Figure 4.1(a), we plotted how the loss changes according to the number of iterations for different learning rates. Observe that, the performance can vary drastically depending on the learning rate. In addition, the optimal learning rate is different depending on the number of iterations (compare the orange line ($\eta = 0.1$) and pink line ($\eta = 0.01$)). When dealing with plaintext, the optimal parameters can be found through cross validation, but this is impossible in MLED because the the weight vector cannot be decrypted during training. On the other hand, our method can obtain a closed-form solution without using gradient descent.



(a) Importance of choosing the appropriate parameters in learning LR

(b) Approximation performance of sigmoid function decreases as possible range of input grows

Figure 4.1: Sensitivity analysis on the hyperparameters in previous privacy preserving LR

Another limitation is that the sigmoid function cannot be evaluated efficiently using HE. Because the low-order Taylor expansion results in a good approximation only in a localized domain, [101] approximated the sigmoid function with a polynomial showing the lowest mean squared error in the domain $[-8, 8]$. Figure 4.1(b) shows the different approximation results according to the domain range when approximated by a 3rd order polynomial. The maximum error between the polynomial and the sigmoid function is 0.11 when the range is $[-8, 8]$ (blue line), which increases to 0.26 when the range is $[-20, 20]$ (red line). Therefore, it is essential to obtain a tight bound for the input value of the sigmoid function, which is not possible for encrypted data. Conversely, our method solves ridge regression on the ciphertext; thus, there is no need to evaluate the sigmoid function.

4.2.2 Framework

In this study, we propose a new framework that can effectively handle classification tasks, while preserving the privacy of private variables. We set two different datasets for the same variables, $\mathcal{D}_1 = \{(\boldsymbol{x}_1^i, y_1^i) | i = 1, \ldots, n_1\}$ and $\mathcal{D}_2 = \{(\boldsymbol{x}_2^j, y_2^j) | j = 1, \ldots, n_2\}$, where $\boldsymbol{x}_2^j = (x_{21}^j, \ldots, x_{2p}^j, [x_{2(p+1)}^j], \ldots, [x_{2(p+\ell)}^j])$ contains ℓ encrypted private variables.

Threat Model The participants of our framework consist of data owners O, a modeler M and a crypto-service provider C. We assume that the participants are honest-but-curious and do not collude, which is widely accepted in MLED studies [144, 128, 8, 130]. Our security goals are as follows:

- Neither M nor C should obtain information on private variables.
- Neither O nor C should obtain information on the learned model.

Protocol The details of our protocol to achieve the security goal are as follows:

- (**Teacher modeling**) M trains a teacher model f_s with an unencrypted dataset $\mathcal{D}_1$, where $\mathcal{D}_1$ is owned by M or publicly available.
- (**Encryption**) C generates keys (pk, sk) and sends pk to O and M. O encrypts the private variables of their dataset $\mathcal{D}_2$ and sends $\mathcal{D}_2$ to M.
- (**Training on encrypted data**) M infers encrypted logit $\mathsf{Enc}(l_2) = f_s(\mathcal{D}_2)$ and evaluates $\mathsf{Enc}(\tilde{l_2}) = \mathsf{Enc}(l_2) + \mathsf{Enc}(\beta)$ through mean matching. M trains the privacy-preserving ridge regression on $\mathcal{D}_2$ and $\mathsf{Enc}(\tilde{l_2})$ and obtains $\mathsf{Enc}(\omega)$.

- (**Decryption**) M generates a random polynomial r and sends $\mathsf{Enc}(\omega + r)$ to C. C decrypts $w+r$, adds a random discrete Gaussian noise e, and sends $w+r+e$ back to M. M subtracts r, and the final weight is obtained as $w + e$.

Similar to [56, 71], the security of our protocol follows directly from the semantic security (against passive adversaries) of the underlying HE scheme. In our protocol, e is added to defend against attack proposed by [113], which, according to [32], with a high probability causes only <1 bits of precision loss.

Our framework consists of three steps, excluding encryption and decryption. The first step is **teacher modeling**, which is the first of our protocol, and the second and third steps are **mean matching** and **ridge regression**, which corresponds with the third protocol.

Teacher modeling In the first step, M trains a classification model with $\mathcal{D}_1$, which mimics the first phase of the knowledge distillation in the first model; then, we extract the soft probability from the target label. However, our method has several major differences from knowledge distillation. First, knowledge distillation aims to train a rather simple model that performs comparably to a complex teacher model, in which extracting the probability is a means to achieve the goal. On the other hand, in our framework the probability plays a more important role. We transform the classification task into a regression problem through the probability. Another difference is that in our method, the teacher model does not need to be a complex model. There are several advantages to using a simple model as a teacher model, which we will demonstrate later.

Meanwhile, we should assume that M should possess an unencrypted $\mathcal{D}_1$ that

can be used in step 1. Considering that M is depicted as a company seeking to profit from their model in many studies [119, 118, 121], there may be individuals who provide their private information to M in return. To be more realistic, we assumed there are relatively few people who publish their private information. Moreover, the underlying distributions of $\mathcal{D}_1$ and $\mathcal{D}_2$ are heterogeneous because they have different features than those who do not want to provide private information. To fit the former setting, we use a simple model that generalizes sufficiently with a small amount of data as the teacher model.

Mean matching In the second step, using the teacher model M infers the logit of $\mathcal{D}_2$. In the case where the teacher model uses LR, the inference is simply an inner product between the model parameter $\boldsymbol{\theta}$ and $\boldsymbol{x}^j$s, which can be efficiently computed with HE using slot-wise rotation (See the Allsum algorithm in [104]). Denoting the teacher model as f_s and logits of $\mathcal{D}_1$ and $\mathcal{D}_2$ as $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$, respectively, we can obtain $\boldsymbol{l}_2$ from $\mathcal{D}_2$ using the teacher model f_s. However, because $\mathcal{D}_1$ and $\mathcal{D}_2$ have different distributions, also $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ might. Therefore, rather than directly using the logit distribution $\boldsymbol{l}_2$, we suggest adding a regularization term β to $\boldsymbol{l}_2$ so that it can consider the difference between two distributions. The β should satisfy:

$$\frac{\sum_i f_s(x_1^i)}{\sum_i y_1^i} = \frac{\sum_i (f_s(x_2^i) + \beta)}{\sum_i y_2^i} \tag{4.3}$$

Now applying the regularization term β, we can obtain the shifted logit $\tilde{\boldsymbol{l}}_2$, where $\tilde{l}_2^i = f_s(x_2^i) + \beta$.

Note that (4.3) can be seen as a simplified version of kernel mean matching reported in [72], which is widely used in domain adaptation studies [111]. Kernel

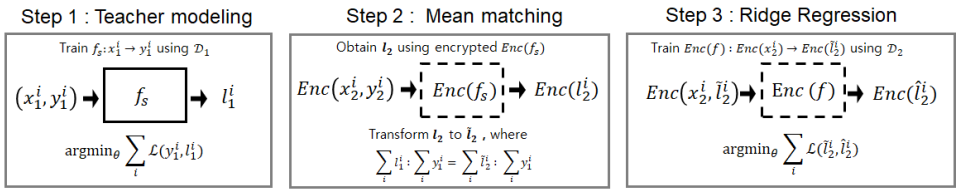


Figure 4.2: Overall framework of our proposed method

mean matching attempts to match the means of the distributions of $\boldsymbol{x}$s in a kernel space. When directly applying mean matching to logits, the distributions of logits are forced to become similar, which has a negative effect on moving the prediction away from the true label. Therefore, we multiplied the logit by a weight, which reflects the distribution of the true label.

Ridge regression Finally, we train a ridge regression on $\mathcal{D}_2$, where the target variable is the shifted logit $\tilde{\boldsymbol{l}}_2$, not $\boldsymbol{y}_2$. Intuitively, using a well-estimated probability enables a training at least as accurate as that achieved when using a binary target; our theoretical results reported in section 4.3 support this claim. By encrypting a small private portion of the entire information, we can train an efficient ridge regression without having to explore the learning parameters, as in Chapter 3.

The ridge regression method can be extended to nonlinear regression by considering linear combination of fixed nonlinear basis functions of the input variables. We omit a detailed formulation, but in case of one private variable, by replacing $\boldsymbol{x}^i = (x_1^i, \ldots, x_p^i, x_s^i)$ with $\boldsymbol{\phi}(\boldsymbol{x}^i) = (\phi_1(\boldsymbol{x}^i), \ldots, \phi_m(\boldsymbol{x}^i), x_s^i)$, the ridge estimate is

$$\hat{\boldsymbol{f}} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda \boldsymbol{I}_{M+1})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y} = \boldsymbol{K}(\boldsymbol{K} + \lambda \boldsymbol{I}_n)^{-1}\boldsymbol{y} \tag{4.4}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{n\times(M+1)}$ whose i-th row is $\boldsymbol{\phi}(\boldsymbol{x}^i)$. Defining the kernel function as

$$K(\boldsymbol{x}^i, \boldsymbol{x}^j) = \phi(\boldsymbol{x}^i)^T \phi(\boldsymbol{x}^j), \tag{4.5}$$

the kernel matrix can be easily obtained as

$$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T = \sum_{i=1}^{p} \boldsymbol{\Phi}_i \boldsymbol{\Phi}_i^T + \boldsymbol{h}\boldsymbol{h}^T = \boldsymbol{\Phi}_{(-s)} \boldsymbol{\Phi}_{(-s)}^T + \boldsymbol{h}\boldsymbol{h}^T = \boldsymbol{K}_{(-s)} + \boldsymbol{h}\boldsymbol{h}^T \tag{4.6}$$

where $\boldsymbol{h}$ is defined the same as in section 3.3. Then all the remaining steps are the same as in section 3.3, with $\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}$ replaced by $\boldsymbol{K}$.

4.3 Theoretical Results

This section provides the core theory behind our proposed methodology. We provide a generalization error bound for the proposed method.

Consider a classification task with an input space $\mathcal{X}$, an output space $\mathcal{Y} = \{0, 1\}$, and hypothesis space $\mathcal{Z} = \mathcal{X} \times [0, 1]$. Let $S = \{\mathbf{z}_i = (\mathbf{x}_i, y_i) \in \mathcal{Z} : \ i = 1, ..., N_s\}$ represent the given open labelled data of N_s samples. In our classification settings, we have two distinct distributions: $(\mathbf{x}, y) \sim \mathcal{D}_S$ according to a sample distribution $\mathcal{D}_S$ over $\mathcal{Z}$ with $y \in \mathcal{Y}$ and $(\mathbf{x}, p) \sim \mathcal{D}_T$ according to a target distribution $\mathcal{D}_T$ over $\mathcal{Z}$ with $p \in [0, 1]$.

Utilizing these concepts and following the notations in [127], a hypothesis $h : \mathcal{Z} \to \Re$ is a scoring function such that we assign to each point $\mathbf{x}$ the class label of the maximum score $h(\mathbf{x}, y)$, that is, $\arg\max_{y \in \mathcal{Y}} h(\mathbf{x}, y)$. The *margin* $\xi_h(\mathbf{z})$ at a sample example $\mathbf{z} = (\mathbf{x}, y) \sim \mathcal{D}_S$ is then defined by

$$\xi_h(\mathbf{z}) = h(\mathbf{x}, y) - \max_{y' \neq y} h(\mathbf{x}, y')$$

Thus, h misclassifies $\mathbf{z}$ iff $\xi_h(\mathbf{z}) \leq 0$.

The generalization error (or risk) of a hypothesis $h \in \mathcal{H} := \{h : \mathcal{Z} \to \Re\}$ in a sample distribution $\mathcal{D}_S$ and a true target distribution $\mathcal{D}_T$ are defined, respectively, by

$$\mathcal{R}_{\mathcal{D}_S}(h) = \mathbb{E}_{\mathbf{z} \sim \mathcal{D}_S}[1_{\xi_h(\mathbf{z}) \leq 0}], \qquad \mathcal{R}_{\mathcal{D}_T}(h) = \mathbb{E}_{\mathbf{z} \sim \mathcal{D}_T}[1_{\xi_h(\mathbf{z}) \leq 0}]$$

We next define the $\mathcal{H}$-divergence as in [12] which measures the discrepancy between

two different distributions as in by

$$D_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) = \sup_{h,h' \in \mathcal{H}} |\mathcal{R}_{\mathcal{D}_S}(h') - \mathcal{R}_{\mathcal{D}_T}(h)|$$

We are now ready to derive the following generalization error risk in our classification setting.

Lemma 4.1. *Let $\mathcal{D}_S$ and $\mathcal{D}_T$ be the sample and the true target distributions, respectively. Then for any hypothesis $h \in \mathcal{H}$, the following inequality holds:*

$$\mathcal{R}_{\mathcal{D}_T}(h) \leq \min_{h' \in \mathcal{H}} \mathcal{R}_{\mathcal{D}_S}(h') + D_{\mathcal{H},\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) \tag{4.7}$$

Proof. For a given $h \in \mathcal{H}$,

$$\begin{aligned} \mathcal{R}_{\mathcal{D}_T}(h) &= \mathcal{R}_{\mathcal{D}_S}(h') + \mathcal{R}_{\mathcal{D}_T}(h) - \mathcal{R}_{\mathcal{D}_S}(h') \\ &\leq \mathcal{R}_{\mathcal{D}_S}(h') + D_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) \end{aligned}$$

which holds for all $h' \in \mathcal{H}$. Therefore, the result follows. ¶ □

Following the notations and definitions in [127], let $S = \{\mathbf{z}_i = (\mathbf{x}_i, y_i) \sim \mathcal{D}_S : i = 1, ..., N\}$ represent the class labeled data of N samples. Then the empirical Rademacher complexity of $\mathcal{H}$ with respect to S is the random variable

$$\hat{\mathfrak{R}}_{\mathcal{D}_S}(\mathcal{H}) = \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \sigma_i h(\mathbf{z}_i) \right] \tag{4.8}$$

where $\boldsymbol{\sigma} = \{\sigma_1, ..., \sigma_N\}$ are independent uniform $\{\pm 1\}$-valued Rademacher random variables. The *Rademacher complexity* of $\mathcal{H}$ is the expectation of the empirical

Rademacher complexity over all samples of size N:

$$\mathfrak{R}_N(\mathcal{H}) = \mathbb{E}_{\mathcal{D}_S}[\hat{\mathfrak{R}}_{\mathcal{D}_S}(\mathcal{H})] = \mathbb{E}_{S\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{H}} \frac{1}{N}\sum_{i=1}^{N} \sigma_i h(\mathbf{z}_i)\right] \tag{4.9}$$

Theorem 4.2. *Let $\mathcal{D}_S$ and $\mathcal{D}_T$ be the sample and the true target distributions, respectively. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following classification generalization bound holds for all hypothesis $h \in \mathcal{H}_\rho = \{(\mathbf{x}, y) \to \boldsymbol{\omega}\cdot(y\mathbf{x}) : \|\boldsymbol{\omega}\|_2 \le 1/\rho,\ \|\mathbf{x}\|_2 \le r\}$:*

$$\mathcal{R}_{\mathcal{D}_T}(h) \leq \frac{1}{N}\sum_{i=1}^{N} \log_{e_0}\left(1 + e^{-2y_i\boldsymbol{\omega}\cdot\mathbf{x}_i}\right) + D_{\mathcal{H}_\rho}(\mathcal{D}_S, \mathcal{D}_T) \tag{4.10}$$

$$+\frac{16r}{\rho\sqrt{N}} + \sqrt{\frac{\log\log_2 \frac{4r}{\rho}}{N}} + \sqrt{\frac{\log\frac{2}{\delta}}{2N}} \tag{4.11}$$

where $e_0 = \log(1+1/e)$.

Proof. We define $\Lambda(\mathcal{H}_1)$ for a scoring function $h \in \mathcal{H}_1$ by

$$\Lambda(\mathcal{H}_1) = \{\mathbf{x} \mapsto h(\mathbf{x}, y) :\ y \in \mathcal{Y},\ h \in \mathcal{H}_1\} = \{\mathbf{x} \mapsto y(\boldsymbol{\omega}\cdot\mathbf{x}) :\ y \in \mathcal{Y},\ \|\boldsymbol{\omega}\|_2 \le 1\}$$

Then for any $0 < \rho < 2r$, by Theorem 9.2 and Theorem 13.2, 13.4 in [127] and some slight modifications adapted to our classification setting, we have the following general margin bound of $\mathcal{R}_{\mathcal{D}_S}(h)$:

$$\mathcal{R}_{\mathcal{D}_S}(h) \leq \frac{1}{N}\sum_{i=1}^{N} 1_{\xi_h(\mathbf{z}_i)\le\rho} + \frac{8}{\rho}\Re_N(\Lambda(\mathcal{H}_1)) + \sqrt{\frac{\log\log_2\frac{4r}{\rho}}{N}} + \sqrt{\frac{\log\frac{2}{\delta}}{2N}}$$

where $e_0 = \log(1+1/e)$.

We next derive a margin upper bound of $\Re_N(\Lambda(\mathcal{H}_1))$. Since $|\boldsymbol{\omega}\cdot y\mathbf{x}| \le \|\boldsymbol{\omega}\|_2\|y\mathbf{x}\|_2 \le r$ by the Cauchy–Schwarz inequality, we have

$$\begin{aligned}\hat{\Re}_N(\Lambda(\mathcal{H}_1)) &= \frac{1}{N}\mathbb{E}_\sigma\left[\sup_{\|\boldsymbol{\omega}\|_2\le 1, y\in\mathcal{Y}} \boldsymbol{\omega}\cdot\sum_{i=1}^N y\sigma_i\mathbf{x}_i\right] = \frac{1}{N}\mathbb{E}_\sigma\left[\sup_{y\in\mathcal{Y}}\|\sum_{i=1}^N y\sigma_i\mathbf{x}_i)\|_2\right] \\ &\le \frac{1}{N}\sum_{y\in\mathcal{Y}}\mathbb{E}_\sigma\left[\|\sum_{i=1}^N y\sigma_i\mathbf{x}_i\|_2\right] \le \frac{1}{N}\sum_{y\in\mathcal{Y}}\sqrt{\mathbb{E}_\sigma\left[\|\sum_{i=1}^N y\sigma_i\mathbf{x}_i\|_2^2\right]} \\ &\le \frac{1}{N}\sum_{y\in\mathcal{Y}}\sqrt{\sum_{i=1}^N\|y\mathbf{x}_i\|_2^2} \le \frac{2r}{\sqrt{N}}\end{aligned}$$

where the third inequality holds by definition of the dual norm. Since

$$\begin{aligned}&\sum_{y\in\mathcal{Y}}\exp(\frac{h(\mathbf{x}_i,y)-h(\mathbf{x}_i,y_i)}{\rho}) = \sum_{y\in\mathcal{Y}}\exp(\frac{y\boldsymbol{\omega}\cdot\mathbf{x}_i - y_i\boldsymbol{\omega}\cdot\mathbf{x}_i}{\rho}) \\ &= \exp(\frac{\boldsymbol{\omega}\cdot\mathbf{x}_i - y_i\boldsymbol{\omega}\cdot\mathbf{x}_i}{\rho}) + \exp(\frac{-\boldsymbol{\omega}\cdot\mathbf{x}_i - y_i\boldsymbol{\omega}\cdot\mathbf{x}_i}{\rho}) = 1+\exp(\frac{-2y_i\boldsymbol{\omega}\cdot\mathbf{x}_i}{\rho})\end{aligned}$$

if we let $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}/\rho$, then $\|\tilde{\boldsymbol{\omega}}\| \le 1/\rho$

$$\frac{1}{N}\sum_{i=1}^N 1_{\xi_h(\mathbf{z}_i)\le\rho} \le \frac{1}{N}\sum_{i=1}^N \log_{e_0}\left(\sum_{y\in\mathcal{Y}} e^{\frac{h(\mathbf{x}_i,y_i)-h(\mathbf{x}_i,y)}{\rho}}\right) = \frac{1}{N}\sum_{i=1}^N \log_{e_0}\left(1+\exp(-2y_i\tilde{\boldsymbol{\omega}}\cdot\mathbf{x}_i)\right)$$

Therefore we have

$$\mathcal{R}_{\mathcal{D}_S}(h) \le \frac{1}{N}\sum_{i=1}^N \log_{e_0}\left(1+\exp(-2y_i\tilde{\boldsymbol{\omega}}\cdot\mathbf{x}_i)\right) + \frac{16r}{\rho\sqrt{N}} + \sqrt{\frac{\log\log_2\frac{4r}{\rho}}{N}} + \sqrt{\frac{\log\frac{2}{\delta}}{2N}}$$

and the result follows from Lemma 4.1. ¶ □

Notice that the first term of the right-hand side of Eq. (4.10) is the loss function for the logistic regression where the conditional probability takes the form of logit

function:

$$\Pr[y = 1|x] = \frac{1}{1 + e^{-2\boldsymbol{\omega}\cdot\mathbf{x}}}$$

As a result, the upper bound of empirical expected target error can be decomposed into two parts: The first term is the empirical source error. In our framework, we try to minimize this term by training a teacher classification model with $\mathcal{D}_1$ in the first step. Then the second term is the $\mathcal{H}$-divergence between the $\mathcal{D}_S$ and $\mathcal{D}_T$ which implies that we should reduce the gap between two distributions to achieve better performance. The mean matching step in our framework corresponds to this part, which is a simplified version of kernel mean matching [72].

4.4 Experiments

In this section, we evaluate our method using various real-world datasets. Through experiments, we argue that our method achieves better classification results compared to the existing methods with a shorter computation time. In addition, we verify that even when the distributions of published and encrypted data are very different, our method can properly correct the difference and maintain the classification performance.

4.4.1 Experimental Setting

Datasets We used five widely used classification datasets from the UCI data repository: The adult income dataset (**Adult**), bank marketing dataset (**Bank**), Wisconsin Breast Cancer dataset (**Cancer**), Pima Indians Diabetes dataset (**Diabetes**), and Australian Credit Approval (**Credit**) dataset [49]. **Adult** and **Bank** datasets are commonly used in fair machine learning studies, thus, we treat variables that can induce social bias, such as gender and age, like private variables. For the other datasets, the variable that had the greatest impact on the classification performance was selected as a private variable. In practical applications, data owners can flexibly set private variables according to their security standards. The explanation for each dataset is provided in Table 4.1.

Experimental Setup All the experiments were performed on a machine equipped with 40 threads of an Intel Xeon E-2660 v3 @2.60GHz CPU processor. We implemented step 1 of our framework with Python 3.6.3, using the LR module in the scikit-learn library. Other steps were implemented with C++, using HEAAN v1.1

Table 4.1: Description of datasets

Dataset	# of attributes	# of samples	Private Variable
Adult	11	30162	'Gender
Bank	10	4521	'Age'
Cancer	9	682	'Bare Nuclei'
Diabetes	8	768	'Blood Pressure'
Credit	10	690	'A5'

[33] for HE. HEAAN is an implementation of the CKKS scheme; a detailed description of the scheme and its parameters are provided in the Appendix. We used privacy-preserving LR depicted in [104] (**LRHE**) as a baseline comparison method because it is the only MLED training model that works within a practical computation time. We do not compare our method to the HE-friendly SVM model reported in [140], because their model assumes that the data owners pre-compute the kernel matrix, which can leak information about the model. Comparing our model to DNN models reported in [118] is not of our interest because their model has a latency that is too high. Note that we trained **LRHE** with all variables encrypted because **LRHE** hardly benefits from partial encryption, as mentioned in section 4.2.1. Indeed, encrypting only private variables for LRHE resulted in time savings of less than 1.2%.

For each dataset, we randomly sampled 20% as test samples, and 20% of the other 80% were treated as plaintext data, which were used for the training of step 1. We encrypted the private variables of the remaining 60% and used them for steps 2 and 3. For CKKS parameters, we used $N = 2^{16}, q_L = 2^{1200}$, and $P = 2^{40}$. The sigmoid function approximation degree for **LRHE** was set to 3 because increasing the degree results in a larger multiplicative depth and less possible number of gradient descents with LHE. In addition, we observed that increasing the degree up to 7 did not

significantly affect the performance of the model. The learning rate for **LRHE** was chosen in $\{0.001, 0.0001, 0.00001\}$ to achieve the best classification performance.

4.4.2 Experimental Results

We compared the methods in terms of their classification accuracy and computation time. The results of the experiments are summarized in Table 4.2. For all datasets, **Ours** achieved higher accuracy and lower latency compared to **LRHE**. In particular, regarding the computation time, **Ours** can reduce latency by 66-68% through efficient computation using private variables. Although the performance of LR using plaintext is not inferior to ours, the performance of **LRHE** degrades owing to the sigmoid approximation and limited iterations of gradient descent. On the other hand, because **Ours** is free from parameter selection, its performance does not decrease compared to the same operation using plaintext.

As an ablation study, we additionally trained step 3 of **Ours** using gradient descent, with all variables encrypted (denoted as **Ours-grad** in Table 4.2). Because the multiplicative depth per iteration of **Ours-grad** is lower than that of **LRHE** owing to not using the sigmoid approximation, we trained **Ours-grad** for eight iterations. The learning rate was set in $\{0.001, 0.0001, 0.00001\}$.The procedure that requires the most computation time in **Ours-grad** is the matrix-matrix multiplication, which causes **Ours-grad** to have a greater latency compared to **LRHE**. However, the computation time per iteration of **LRHE** is higher than that of **Ours-grad**, and the result of the matrix-matrix multiplication can be reused through the iterations after calculating once. Therefore, it is implied that as the number of iterations increases, **Ours-grad** will be more efficient than **LRHE**. In addition, the classification

Table 4.2: Classification Results on five datasets.

Dataset	Accuracy (%)			Computation time (sec)			Time per iteration (sec)	
	LRHE	**Ours**	**Ours-grad**	**LRHE**	**Ours**	**Ours-grad**	**LRHE**	**Ours-grad**
Adult	81.913	**83.123**	82.759	306.061	**102.25**	1488.65	58.372	**55.786**
Bank	89.712	89.823	**89.934**	248.854	**82.315**	1095.44	47.528	**43.184**
Cancer	86.957	**90.580**	**90.580**	225.286	**73.392**	968.019	42.419	**37.791**
Diabetes	71.429	75.325	**76.623**	184.733	**61.183**	700.522	35.038	**30.927**
Credit	97.794	98.453	**98.529**	201.763	**65.344**	828.833	38.415	**34.757**

performance of **Ours-grad** is similar to or better than that of **Ours**, which means that **Ours-grad** converges better by allowing more iterations than **LRHE**. Therefore, even in situations where the partial encryption of private variables is limited, our method has an advantage over the existing methods.

Effectiveness of mean matching In this section, we verify that the mean matching in step 2 of our method is a simple but effective way to mitigate performance degradation due to the difference between distributions of unencrypted and encrypted data. Here, we do not randomly divide the dataset, instead divide it referring to the value of a certain variable. **Diabetes** dataset, for example, was divided according to whether the value of 'Glucose' variable was greater than (or less than) its median. Figure 4.3(a) plots the dimension reduction result for **Diabetes** dataset using t-stochastic neighborhood embedding and confirms that the two subsets (X_1 and X_2) have very separate distributions compared to each other. We sampled 20% of the entire dataset from X_1 and trained step 1 with the samples. 80% of X_2 was used for steps 2 and 3, with the remaining 20% of X_2 used as the test samples.

Table 4.3 summarizes the experimental results for three datasets: **Adult**, **Diabetes** and **Credit**. For the other two, there were no variables that could properly divide them. The test accuracy of the LR model trained with X_1 was lower than that

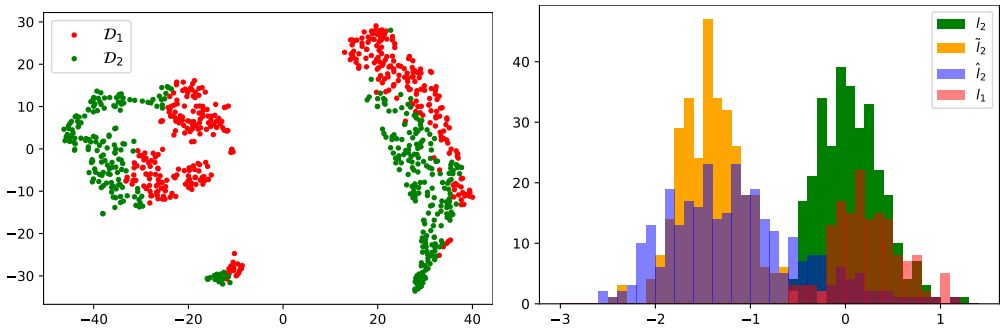
Table 4.3: Ablation study for mean matching. **Criteria** refers to the criteria for dividing the datasets.

Dataset	Criteria	Accuracy (%)			
		LR with X_1	Ours-without mean matching	Ours	LRHE
Adult	X_1 : 'Marital' = 0 X_2 : 'Marital' = 1	63.330	63.330	69.471	68.690
Diabetes	X_1 : 'Glucose' > 117 X_2 : 'Glucose' ≤ 117	29.487	29.487	80.769	80.769
Credit	X_1 : 'A4' = 0 X_2 : 'A4' = 1	20.833	20.833	79.167	79.167

of the **LRHE** trained with X_2 because the domains of X_1 and X_2 were different, and the test set was sampled from X_2. The performance of **Ours** without mean matching was not different from that of the LR with X_1. However, with mean matching the classification accuracy of **Ours** was raised to the same level as that of **LRHE**. Even when the accuracy of LR with X_1 was 50% or lower than that of **LRHE**, applying mean matching completely recovered the classification performance of **Ours**.

To verify why the mean matching of our method works well, in Figure 4.3(b), we plotted the histogram of logits inferred by LR models. In the figure, the red and blue histograms represent the distributions of logit of X_1 for LR learned with X_1 ($=\boldsymbol{l}_1$) and logit of X_2 for LR learned with X_2, respectively. Thus, the blue histogram can be seen as the distribution of the logit of X_2 when it is classified "correctly". However, because the model is biased toward the distribution of X_1 when trained with X_1, the distribution of the logit of X_2 ($=\boldsymbol{l}_2$) is also biased toward the distribution of the logit of X_1, as shown in the green histogram. Therefore, mean matching plays the role of shifting the biased distribution (orange histogram) to fit the correct distribution.

Extension to nonlinear methods We validated the nonlinear method on the same datasets . To give nonlinearity to the teacher model, we trained a neural



(a) Visualization of $\mathcal{D}_1$ and $\mathcal{D}_2$ with different distributions.

(b) Logit distribution of $\boldsymbol{l}_1$ (red), $\boldsymbol{l}_2$ (green), $\tilde{\boldsymbol{l}}_2$ (orange). The blue histogram refers to the distribution of logit of X_2 when trained with X_2.

Figure 4.3: Effectiveness of Mean Matching

Table 4.4: Results for our nonlinear regression model

Dataset	**Accuracy (%)**				**T-test (Ours-nonlinear > Ours-linear)**	
	NN (step 1)	**LRHE**	**Ours-linear**	**Ours-nonlinear**	**t-statistic**	**p value**
Adult	83.90	80.106	81.217	**83.803**	6.412	<0.001
Bank	87.72	87.058	87.279	**87.943**	1.093	0.144
Cancer	94.85	94.118	**94.853**	**94.853**	-0.323	0.625
Diabetes	75.68	70.130	75.325	**76.623**	5.942	<0.001
Credit	88.41	86.232	86.232	**89.623**	7.911	<0.001

network which consists of three fully-connected layers as the teacher model. Each layer reduces the data dimension to 8, 4, 1, respectively, and a sigmoid function was taken after the last layer. After the first two layers, we used square activation instead of ReLU for an HE-friendly inference. The step 3 of our method was trained with the kernel Ridge regression demonstrated above. The results are summarized in Table 4.4. It is shown that the nonlinear extension works well with every dataset, achieving at least as high accuracy as our original method. Except for **Adult** and **Cancer** dataset, the nonlinear extension showed significantly better ($p < 0.001$) accuracy compared to the original method. This is because linear Ridge regression

cannot represent the nonlinearity obtained from the nonlinear teacher model well. In the case of **Cancer**, it seems that the linear boundary is close to the optimum, so the same performance is obtained even if nonlinearity is not introduced. **Ours-nonlinear** will perform better if we use a more complex teacher model or a dataset where the linear teacher model performs poorly.

4.5 Chapter Summary

We proposed an efficient HE-friendly classification method, that protects both the users' private information and secrecy of the model. We trained a ridge regression model for the logit instead of logistic regression, which has a closed-form solution and is free from parameter search. To extract logit values from a binary label, we trained a teacher model on unencrypted data that can output logit, similar to that of knowledge distillation. Owing to encrypting only private variables that require a high level of security instead of encrypting all information, our method can achieve higher efficiency and training stability. Our algorithm is HE scheme-free; it can bring efficiency when implemented with any widely used HE schemes.

Although our study aims to extend the existing machine learning method in the privacy-preserving direction, our model can be corrupted by malicious participants because our security model is limited to semi-honest, non-colluding participants. Designing a secure system for weaker security assumptions should be conducted in future studies.

Chapter 5

Homomorphic-encryption-friendly Evaluation for Support Vector Clustering

5.1 Problem Statement

Recently, machine learning technologies have successfully solved real-world problems in various fields, including biomedical and financial applications. Sensitive data such as personal data, biometric data, medical information, and financial information can be used to construct high-quality machine learning models. Therefore, moral and legal issues about data protection and privacy use have received attention, and the conflict between data utilization and protection needs to be addressed.

Fully homomorphic encryption (FHE), which enables numerical operations on encrypted data, is considered to be a promising direction that satisfies data utilization and protection [61, 33]. Privacy-preserving machine learning algorithms with FHE have been proposed to train supervised models and evaluate the models on the encrypted domain, where encrypted data can be transmitted to the model without revealing the original data [71, 140]. However, implementing machine learning algorithms with FHE involves much slower computations and much larger data storage than implementing the same algorithms on the plaintext domain. Implementing machine learning algorithms without considering the operational characteristics of

FHE can worsen the problems and degrade the performance.

Clustering is a representative unsupervised learning task widely used in areas including image segmentation, information retrieval, and marketing. Clustering algorithms partition given instances into a set of subgroups called clusters depending on their similarity (or distance). Clustering on the encrypted data can be more complicated than classification or regression because the shape and the number of clusters are unknown. Clustering algorithms such as k-means clustering and mean shift clustering methods have been implemented on encrypted data but lack the performance of complex and non-convex data [4, 36]. In contrast, the support vector clustering (SVC) algorithm can capture the complex shape of clusters by labeling the support of data distribution based on the support vector domain description (SVDD) [110, 102].

In this chapter, we propose a privacy-preserving evaluation for SVC. Our work aims to implement an efficient SVC inference on the encrypted domain, where it can capture the complex data distribution with a support function and assign the most appropriate cluster for a new test data. Our algorithm enables robust SVC labeling for test data on an encrypted domain without decryption by configuring the entire procedure as a homomorphic operation. In the experiments, six datasets were used to evaluate the performance of clustering algorithms on the encrypted domains.

5.2 Background

5.2.1 CKKS scheme

In this chapter, we used CKKS scheme that supports the approximate computation of real numbers, where a small error is added to the plaintext vector after decryption [33]. CKKS can provide efficient floating-point operations at the expense of a bounded loss of precision. A complex vector (plaintext) can be encoded into a ring element and encrypted into a single ciphertext, and the slot rotation of the vector on the encrypted domain enables an efficient parallel computation of the ciphertexts. Because of efficient computation, many machine learning algorithms based on CKKS have been proposed for real-world applications [36, 140].

Initializing CKKS scheme, some parameters are determined to achieve a targeted level of security, including N, initial ciphertext modulus $\log q_L$ and scaling factor p, where N is related to the ciphertext space and the number of plaintext slots ($= N/2$). The parameters $\log q_L$ and p determine the precision of the calculations and the number of operations without bootstrapping. Since the error in the evaluated ciphertext grow rapidly with multiplication compared to other operations, it is necessary to rescale the ciphertext after multiplication to control the magnitude of the error with decreasing the ciphertext modulus.

The bootstrapping procedure allows the evaluated ciphertext to be refreshed by homomorphically evaluating the Decrypt function with increasing the ciphertext modulus. In the case of CKKS, the computational cost of bootstrapping increases with the number of plaintext slots with $O(\log N/2)$ [31]. Although bootstrapping of CKKS is more efficient than other FHE schemes, it is still the most expensive part

of CKKS. Therefore, it is essential to consider the trade-off between the efficiency of parallel operation and the cost of bootstrapping.

Algorithm 3 demonstrates the procedures used for key generation, encryption and decryption. As an important feature of CKKS, it has a unique method of encoding a plaintext vector into a ring element prior to encryption. Through Encode, CKKS supports encryption for complex vectors, and naturally enables a SIMD operation as well. Decode is almost the same as the inverse of Encode and is used after Decrypt.

Algorithm 3 Basic Procedures for CKKS scheme

1: $\mathsf{KeyGen}(1^\lambda) \rightarrow (\mathsf{sKey}, \mathsf{pKey}, \mathsf{evKey})$ ▷ Key generation
2: $\mathsf{Encode}(\vec{z}) \rightarrow m \in \mathcal{R}$ ▷ Complex vector to polynomial ring
3: $\mathsf{Decode}(m) \rightarrow \vec{z} \in \mathbb{C}^{N/2}$ ▷ Polynomial ring to complex vector
4: $\mathsf{Encrypt}(\mathsf{pKey}, m) \rightarrow c$ ▷ Encryption
5: $\mathsf{Decrypt}(\mathsf{sKey}, c) \rightarrow m$ ▷ Decryption

Algorithm 4 presents the procedures for addition and multiplication operations. By combining these procedures, the evaluation of an arbitrary function is performed. Whereas Add and Mult refer to the addition and multiplication between ciphertexts, ConstAdd and ConstMult refer to operations between a ciphertext and a constant polynomial. In any method that supports parallel operation as well as CKKS, since these operations are performed in units of slots, as many operations as the number of plaintext slots can be performed at a time. The scheme additionally includes Rotate and Rescale procedures. Rotate is slot-wise rotation, and is are particularly useful when adding values in the same ciphertext, and enable efficient parallel operations. Rescale is appended after every ConstMult or Mult to manage the magnitude of the error. Rescale reduces the ciphertext modulus, which limits the number of operations performed without bootstrapping.For a more detailed description of the scheme and

the algorithms, refer to [33]. It should emphasized that most operations are included in the general scheme described in Section 2.1.

Algorithm 4 Operation Procedures for CKKS scheme

1: $\mathsf{ConstAdd}(c, v \in \mathcal{R}) \rightarrow c_{\mathsf{add}}$ ▷ Addition between ciphertext and plaintext
2: $\mathsf{Add}(c, c') \rightarrow c_{\mathsf{add}}$ ▷ Addition between ciphertexts
3: $\mathsf{Sub}(c, c') \rightarrow c_{\mathsf{sub}}$ ▷ Subtraction between ciphertexts
4: $\mathsf{ConstMult}(c, v \in \mathcal{R}) \rightarrow c_{\mathsf{mult}}$ ▷ Multiplication between ciphertext and plaintext
5: $\mathsf{Mult}(\mathsf{evKey}, c, c') \rightarrow c_{\mathsf{Mult}}$ ▷ Multiplication between ciphertexts
6: $\mathsf{Rotate}(\mathsf{c}, j) \rightarrow c_{\mathsf{Rotate}}$ ▷ Rotation of a ciphertext for j slots

5.2.2 SVC

Support-based clustering starts with estimating the support function of data distribution obtained by the SVDD, Gaussian process clustering, or kernel density estimation. In this study, we used SVDD with the Gaussian kernel to obtain the support function as follows:

$$s(\mathbf{x}) = 1 - 2\sum_{i=1}^{N_v} \beta_i e^{-\delta\|\mathbf{x}-\mathbf{x}_i\|^2} + \sum_{i=1}^{N_v}\sum_{j=1}^{N_v} \beta_i\beta_j e^{-\delta\|\mathbf{x}_i-\mathbf{x}_j\|^2} \tag{5.1}$$

where $\delta > 0$ is the width parameter for the Gaussian kernel and N_v denotes the number of support vectors (SVs). The support function (5.1) can be used to partition the data space into basin cells by constructing the following dynamical system:

$$\frac{d\mathbf{x}}{dt} = -\nabla s(\mathbf{x}). \tag{5.2}$$

A stable equilibrium vector (SEV) of the system (5.2) is an equilibrium state where all the eigenvalues of Hessian $\nabla^2 s(\mathbf{x})$ are positive. Then, the basin cell $B(\mathbf{s}_i)$ is defined as the closure of the set of all the data points that converge to a SEV $\mathbf{s}_i$

following the system (5.2). Some SEVs are connected to constitute clusters using the characteristic of the dynamical system (5.2), and the points in a basin cell are labeled with the cluster label of the corresponding SEV [110, 102, 141]. SVC can be inductive and have an efficient and stable inference phase with using the system (5.2) [110, 141].

5.3 Proposed Method

In user-server scenarios, the inference phase of the clustering algorithm with FHE can provide to partition the given encrypted instances while protecting user's privacy from curious servers. In this letter, we present the inference phase of the SVC algorithm with FHE because SVC is naturally inductive and has a stable inference. For SVC, most inference methods are based on the system (5.2) that a test point follows to find the SEV. In particular, the simplest inference method is to allocate the test points to the cluster of the closest SEV. However, this inference cannot capture the complex clusters because it ignores the intrinsic data distribution. Therefore, we present an elaborate inference algorithm of SVC which utilizes the support function (5.1) of the data distribution. We addressed the challenge of dealing with HE-unfriendly operations of SVC inference while balancing efficiency and accuracy.

We propose a HE-friendly evaluation that utilizes the basin induced from the system (5.1) to stabilize the inference. Our algorithm consists of two parts. In the first part, we use the dynamical system (5.2) to move the given test data to more stable points. We can postulate that the boundary regions between two different basins are the most unstable, whereas the SEVs are the most stable points. Thus, our algorithm makes the points evade the boundary region by applying the system (5.2) which converges to SEVs. In the second part, our algorithm assigns the cluster label by finding the SEV closest to the point obtained after the first part. The procedure of our proposed method is presented in Algorithm 5.

In Algorithm 5, we need to implement the calculation of the gradient of the support function $\nabla s(\mathbf{x})$ homomorphically. When using the Gaussian kernel, $\nabla s(\mathbf{x})$

Algorithm 5 HE Friendly Evaluation of Support-based Clustering

Input : Test data $\{\mathbf{x}_i\}_{i=1}^{N_{test}}$, SEVs $\{\mathbf{s}_i\}_{i=1}^{N_S}$, SVs $\{\mathbf{v}_i\}_{i=1}^{N_V}$, support function s, number of iteration T, learning rate η

Output : Clustering label $\{l_i\}_{i=1}^{N_{test}}$

1: **for** $i = 1$ to N_{test} **do**
2: $\quad \mathbf{x}_i^0 \leftarrow \mathbf{x}_i$
3: $\quad$ **for** $t = 1$ to T **do**
4: $\quad\quad \mathbf{x}_i^t \leftarrow \mathbf{x}_i^{t-1} - \eta \cdot \nabla s(\mathbf{x}_i^{t-1})$
5: $\quad$ **end for**
6: $\quad$ Find the nearest SEV ($\mathbf{s}_i$) of $\mathbf{x}_i^T$, and label $\mathbf{x}_i$ as l_i, the label of $\mathbf{s}_i$.
7: **end for**

can be directly calculated as:

$$\nabla s(\mathbf{x}) = 4\delta \sum_{j=1}^{N_v} \beta_j e^{-\delta \|\mathbf{x}-\mathbf{v}_j\|^2} \cdot (\mathbf{x} - \mathbf{v}_j). \tag{5.3}$$

CKKS is used to efficiently implement the approximate computation of real numbers on encrypted data, which supports homomorphic additions and multiplications. However, it requires polynomial approximations for non-polynomial operations. The gradient (5.3) contains the exponential function and the distance between a test point and the SVs. After the first part has been found without decryption, finding the nearest SEV consists of two components: calculating the distances between a test point and the SEVs and finding the minimum distance between them. The exponential function is approximated with a Taylor expansion. To obtain the min-index of the distances, we used the iterative algorithm in [35]. However, the appropriate packing strategy is needed to improve the computational efficiency of calculating the distances and finding the min-index.

We design the plaintext packing to avoid bootstrapping. For simplicity, we assume that all vectors, including test data, SVs, and SEVs, are d-dimensional row

vectors. Figure 5.2 shows the structures of the packed plaintexts, where a plaintext vector will be encrypted into a ciphertext. Note that CKKS supports addition, element-wise multiplication, and right-shift and left-shift rotations for plaintext vectors on the encrypted domain. We designed the plaintext vector to efficiently calculate the distances using these operations because both parts of Algorithm 5 involve the calculation of the distances. The test samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_{test}}$ are repeated as many as N_V, while each instance of SVs $\mathbf{v}_i$ and SEVs $\mathbf{s}_i$ are repeated as many as the number of test data N_{test} as in Figure 5.2. For SEVs, N_{test} times repeated vectors are additionally repeated N_V/N_S times. The plaintexts become $dN_{test}N_V$-dimensional vectors which consist of d-dimensional sub-vectors. In this way, subtracting all SVs or SEVs from all test data can be done with only one operation. In addition, $\beta = [\beta_1, \ldots, \beta_{N_V}]$ were packed in the same way as SVs, where β_j's were repeated d times to constitute a vector like $\mathbf{v}_j \in \mathbb{R}^d$. Depending on the number of test samples N_{test} and SVs N_V, we can use multiple ciphertexts for a plaintext vector and parallelize the homomorphic operations.

Finally, we can efficiently compute the difference $\mathbf{x}_i - \mathbf{v}_j$ for obtaining the gradient (5.3) and $\mathbf{x}_i - \mathbf{s}_j$ for finding the closest SEV. After constituting all necessary ciphertexts, we can compute the gradient using homomorphic operations without decryption by replacing the entire operations with the polynomial operations. When finding the nearest SEV, we use the min-index algorithm whose output has a reciprocal of the number of minimal elements for the indices of the minimal elements and 0 for the other indices as in [36]. Because the min-index algorithm on the encrypted domain involves an approximate error, the outputs of the minimal elements can be indistinguishable from 0s with the errors when the number of minimal elements is

large. However, the gradient phase of our algorithm induces the test samples near the boundary to move to the SEV, resulting in more accurate homomorphic results for the min-index algorithm. Figure 5.1 illustrates the change of test points after one gradient descent iteration. We can notice that the test points move toward the SEVs of their basins and away from the other SEVs simultaneously. It can affect the clustering performance for complex data distribution.

Table 5.1: Summarization of the datasets

Dataset	Instances	Attributes	Clusters	Convexity
Hepta	212	3	7	convex
Tetra	400	3	4	convex
Lsun	400	2	3	convex
TwoDiamonds	800	2	2	convex
Target	758	2	2	non-convex
Chainlink	1000	3	2	non-convex

5.4 Experiments

5.4.1 Experimental Setting

We evaluated the proposed method on six datasets shown in Table 5.1 from the fundamental clustering problems suite [168]. We compared clustering performance in terms of the adjusted Rand index (ARI) metric and computation time with HE-friendly mean shift clustering (Meanshift) and k-means clustering (KMeans), which are currently the state-of-the-art clustering models used with FHE. The ARI measures the similarity between two data partitions and has a value between 0 and 1, where 1 represents a perfect agreement between two data partitions. To verify how the error included in ciphertext affects the clustering results, we compared the results of Algorithm 5 to encrypted data (ARI-enc) and unencrypted data (ARI-no). We implemented the evaluation of KMeans and Meanshift, following [36].

We used an Intel Xeon CPU E5-2660 v3 @2.60GHZ processor. We set $\log q_L = 1200$, $\log p = 30$, $N = 2^{16}$ and $\delta = 2$ for all datasets. We conducted the experiments with different test rates in $\{0.2, 0.5, 0.9\}$, which means the ratio between the number of test data and the number of training and test data. Training data was sampled five times for each experiment to measure the average performance. For Algorithm

5, we used one iteration step of (5.2), where the learning rate was set to 0.5 for the Chainlink dataset and 0.8 for the other datasets.

5.4.2 Experimental Results

Table 5.2 shows the clustering results. KMeans showed almost the same computation time for all experiments since a plaintext vector is encrypted into a single ciphertext. In contrast, except Hepta and Lsun, the proposed method showed the longest execution time because it needed to split a plaintext into multiple ciphertexts. The total execution time is highly dependent on the number of ciphertexts, which is related to the number of SVs for the proposed method, to the number of modes for Meanshift, and to the number of clusters for Kmeans. Therefore, we can reduce the computational cost if the SVC algorithm obtains a sparse support function (5.1) with few support vectors.

For Hepta and Tetra datasets, which have simple and convex distributions, all the models showed good performance. For all the other datasets, our method showed the highest ARI value regardless of the test rate. The other algorithms showed poor performance, especially for the Target and Chainlink datasets that have non-convex data distributions, while our model achieved high ARIs. For the Lsun dataset that consists of rectangular clusters with different lengths and widths, the ARI values of the KMeans and Meanshift algorithms for encrypted and unencrypted data are different significantly, whereas SVC always showed consistent results even after encryption. These results demonstrate that by pulling the data point at the boundary toward the center of the basin cell, SVC can attain robustness against the error that can occur from homomorphic operations.

We additionally conducted t-test between ARI scores of the proposed method and the compared methods. Table 5.3 demonstrates the results for five datasets except **Hepta** where all the methods always achieved ARI= 1. Except for **Tetra** where SVC showed the worst clustering perfomance, SVC showed siginificantly better ($p < 0.001$) clustering results than the other methods. For **TwoDiamonds** dataset, even though SVC showed the highest ARI score, the difference between SVC and KMeans was not significant enough.

Figure 5.3 shows the result of our investigation of the difference in clustering performance by presenting the clustering results for three data sets. Figure 5.3(a) and Figure 5.3(b) illustrate that KMeans and Meanshift failed to capture the clusters, especially for the data points at the boundary of the divided region. However, our method correctly allocated the clusters because of the robustness of our method, as stated in the above paragraph. In the case of datasets with nonconvex distributions such as Target and Chainlink, we found that KMeans essentially did not reflect the non-convex distribution, and Meanshift was not able to connect the clusters into a single cluster. On the other hand, SVC completely clustered the Target dataset, and for the Chainlink dataset, almost all data points were properly clustered except for the points located where two clusters are adjacent on another.

5.5 Chapter Summary

We proposed a privacy-preserving evaluation algorithm of SVC with fully homomorphic encryption. Our model enables the allocation of the cluster label for new test data without decryption, improving the clustering performance for non-convex data, and provides robustness of data points near the boundary. The experimental results show that the proposed method effectively clusters encrypted data with various distributions in a realistic amount of time. In the future, we can improve the computational cost of our algorithm by training a sparse SVC model and parallelizing computationally expensive operations when using multiple ciphertexts.

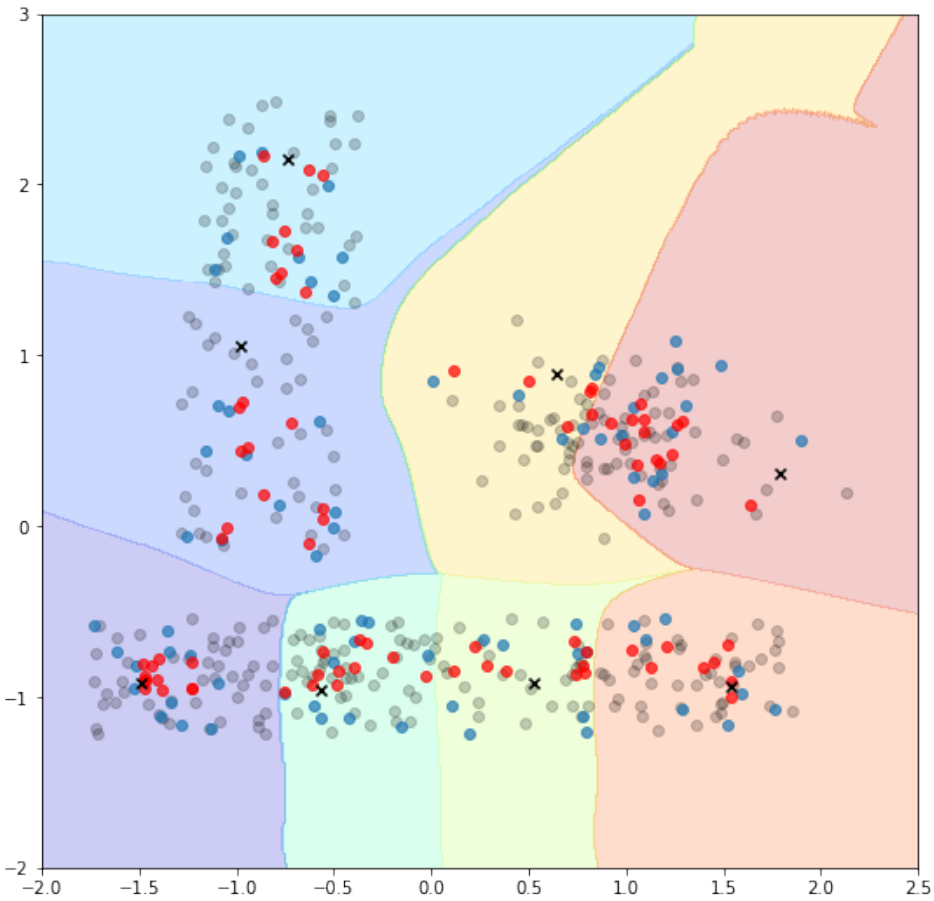


Figure 5.1: Illustration of gradient step for Algorithm 5, where the gray points represent training data, the blue points represent test data, the red points represent the data point after each gradient step, and each 'x' point represents the corresponding SEV of the basins region with different colors.

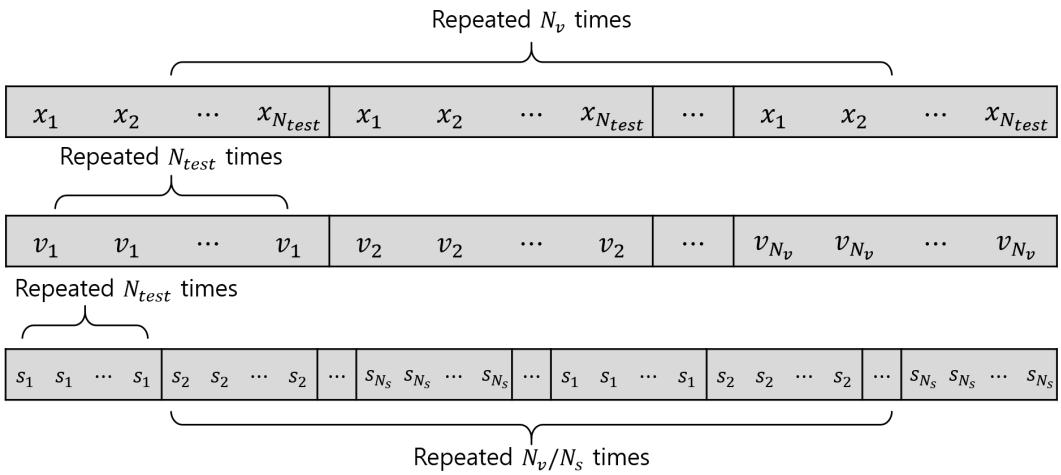


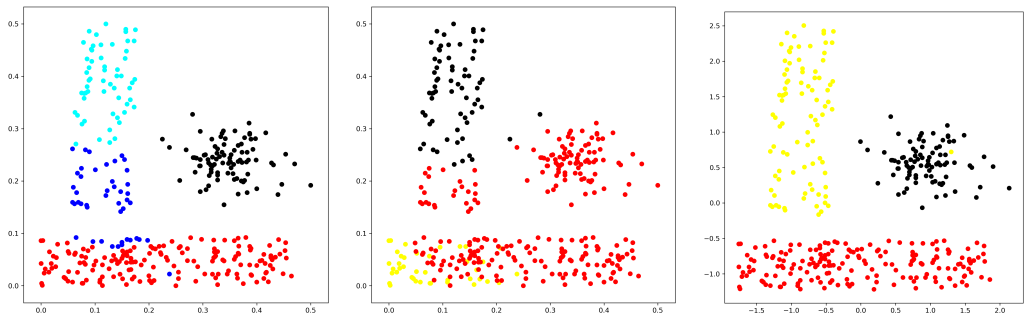
Figure 5.2: Description of the packing method

Table 5.2: Comparison of the results of three algorithms on FCPS datasets.
For each experiment the highest ARI value is highlighted in bold.

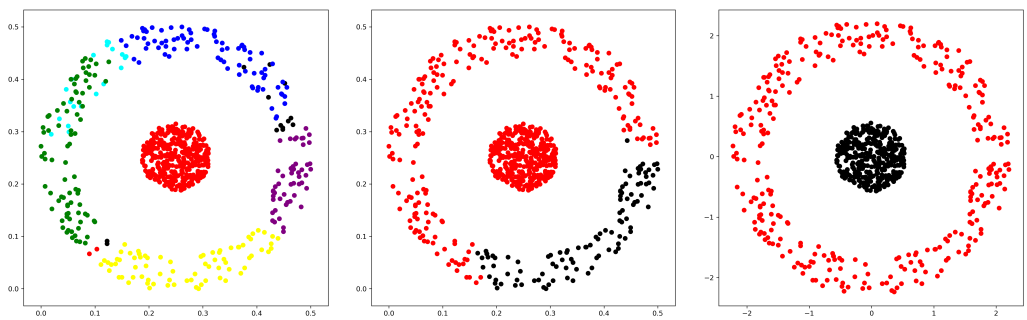
Dataset	Test rate	KMeans		Meanshift		SVC	
		ARI-enc ARI-no	time(s)	ARI-enc ARI-no	time(s)	ARI-enc ARI-no	time(s)
Hepta	0.2	1 1	16.967	1 1	17.332	1 1	30.402
	0.5	1 1	17.001	1 1	16.994	1 1	28.360
	0.9	1 1	17.025	1 1	17.028	1 1	27.239
Tetra	0.2	**1** **1**	16.162	0.957 0.970	16.962	0.927 0.927	57.585
	0.5	**1** **1**	16.505	0.973 0.973	17.383	0.947 0.947	113.896
	0.9	**0.994** **0.994**	16.110	0.915 0.915	17.033	0.961 0.961	109.437
Lsun	0.2	0.351 0.417	15.386	0.629 0.613	16.543	**0.875** **0.875**	25.673
	0.5	0.346 0.400	15.328	0.611 0.651	16.494	**0.931** **0.931**	25.837
	0.9	0.342 0.400	15.409	0.663 0.734	16.171	**0.993** **0.993**	25.368
Two Diamonds	0.2	0.897 0.897	14.978	0.812 0.812	15.476	**0.995** **0.995**	26.147
	0.5	0.891 0.891	15.085	0.842 0.842	15.404	**0.934** **0.934**	52.469
	0.9	0.879 0.879	14.640	0.777 0.777	15.534	**0.923** **0.923**	102.129
Target	0.2	0.105 0.105	14.694	0.619 0.623	17.055	**1** **1**	26.375
	0.5	0.119 0.119	14.822	0.620 0.620	17.331	**1** **1**	53.773
	0.9	0.121 0.121	14.857	0.627 0.627	17.185	**1** **1**	104.359
Chain-link	0.2	0.094 0.094	15.639	0.235 0.231	18.890	**0.853** **0.853**	111.958
	0.5	0.098 0.098	15.465	0.228 0.229	37.062	**0.887** **0.887**	223.896
	0.9	0.068 0.068	15.548	0.318 0.318	15.513	**0.967** **0.967**	220.380

Table 5.3: T-test results between the proposed method and the compared method with test rate= 0.2.

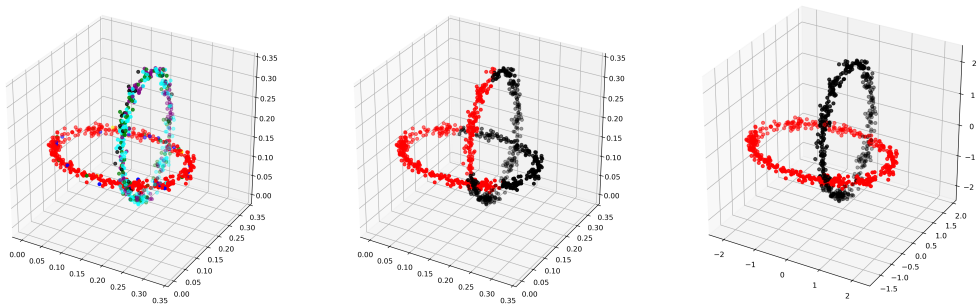
Dataset	**SVC >Meanshift**		**SVC >KMeans**	
	t-statistic	**p value**	**t-statistic**	**p value**
Tetra	-5.874	0.999	-15.032	0.999
Lsun	5.946	<0.001	15.058	<0.001
Two Diamonds	6.514	<0.001	2.895	0.006
Target	31.609	<0.001	49.129	<0.001
Chainlink	27.537	<0.001	29.286	<0.001



(a) Lsun-Meanshift (b) Lsun-KMeans (c) Lsun-SVC



(d) Target-Meanshift (e) Target-KMeans (f) Target-SVC



(g) Chain-Meanshift (h) Chain-KMeans (i) Chain-SVC

Figure 5.3: Visualization of clustering results for Lsun, Target and Chainlink datasets. Each estimated cluster is color-coded.

Chapter 6

Differentially Private Mixture of Gaussians Clustering with Morse Theory

6.1 Problem Statement

Recently, it was widely known that machine learning algorithms can reveal sensitive information about individuals used for training. Among various defense methods against those threats, differential privacy (DP) provides a mathematical guarantee which can prevent the leakage of the personal information from the outputs of the algorithms. In particular, DP has the advantage of being able to quantify the level of privacy protection. To achieve DP, randomness should be added to the original algorithm, which generally degrades the performance. Therefore, studies on differentially private machine learning aim to preserve the performance of the algorithms as high as possible at the same level of privacy protection [68].

Clustering, a representative unsupervised learning technique, connects similar data points into the same cluster. Clustering is widely applied in various real world applications, such as marketing [84], fraud detection [148], recommendation [155], and image segmentation [41]. Despite its importance, there have been few studies on differentially private clustering, compared to supervised learning. Existing studies focus on relatively simple methods such as k-means clustering [160, 182, 174] or

mixture of Gaussians [138, 95, 142]. Those methods are simple yet effective, and the amount of randomness to ensure DP for the methods can be estimated effectively. However, they have a limitation that they cannot express complex, nonconvex cluster structures well.

[107] applied Morse theory to improving the clustering performance of non-private support vector based clustering algorithms. By utilizing Morse theory, their method has a strength in capturing arbitrary cluster shapes. However, nonlinear support vector machines suffer in extending to a differentially private algorithm, because calculating the kernel function in their inference phase requires a set of support vectors, which is a part of training data.

To address the limitations of the previous works, in this study we propose a method to extend the existing differentially private clustering algorithms to capture more sophisticated clusters by applying Morse theory. Specifically, we utilize Gaussian mixture models which does not require any information about training data after the density function is estimated. From the Gaussian mixture density function obtained as a result of differentially private clustering, we build an associated dynamical system and construct a weighted graph, with which we can connect the small sub-clusters. By dividing a complex cluster into multiple convex sub-clusters, our method can effectively infer the original shape of the cluster by merging the sub-clusters. The theoretical results demonstrate that the dynamical processing is essentially DP-friendly, hence the proposed method does not incur additional privacy loss on the existing clustering method. Also, it is proven that our method is inductive and can achieve any desired number of clusters.

The contributions of the study are summarized as:

- By applying Morse theory, the proposed method can detect more complex clusters compared to existing DP-clustering methods. We show that the hierarchical procedure based on Morse theory is DP-friendly and thus does not incur any additional privacy loss. Also, we prove that for a density function expressed as a mixture of Gaussians, our method can capture any desired number of clusters.

- We propose a dynamical process associated with Gaussian mixture model, and prove that it is inductive and does not require retraining for clustering new data. We also prove that the applied inductive dynamical processing is DP-friendly.

- The experimental results on various real-world datasets show that the proposed method improves the clustering performance of the existing methods.

6.2 Background

6.2.1 Mixture of Gaussians

For a D-dimensional data point $\boldsymbol{x}$, the density function of mixture of Gaussians (MoG) model can be written as a linear combination of normal distributions:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{6.1}$$

where $\boldsymbol{\mu}_k \in \mathbb{R}^D, \boldsymbol{\Sigma}_k \in \mathbb{R}^{D\times D}$ are the mean and the covariance of each normal distribution. MoG is demostrated as a latent variable model, by introducing a discrete variable $\boldsymbol{z} \in \{0,1\}^K$ whose k-th element $z_k = 1$ if $\boldsymbol{x}$ belongs to k-th cluster and 0 else. Then $\pi_k \in \mathbb{R}$ can be interpreted as the marginal distribution of $\boldsymbol{z}$, such that $\pi_k = p(z_k = 1)$.

Like other latent variable models, the likelihood of MoG is maximized via EM algorithm. In E-step, the responsibility $\gamma(z_{nk}) = p(z_{nk} = 1|\boldsymbol{x}_n)$ is estimated according to the current parameters. In M-step, The parameters are updated according to the responsibility. The two steps are repeated until convergence as follows:

- (E-step) The responsibility $\gamma(z_{nk}) = p(z_{nk} = 1|\boldsymbol{x}_n)$ is estimated according to current parameters:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^{K} \pi_l \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}. \tag{6.2}$$

- (M-step) The parameters are updated according to the responsibility:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n \tag{6.3}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \tag{6.4}$$

$$\pi_k = \frac{N_k}{N} \tag{6.5}$$

  where N is the number of data points and $N_k = \sum_{n=1}^{N} \gamma(z_{nk})$.

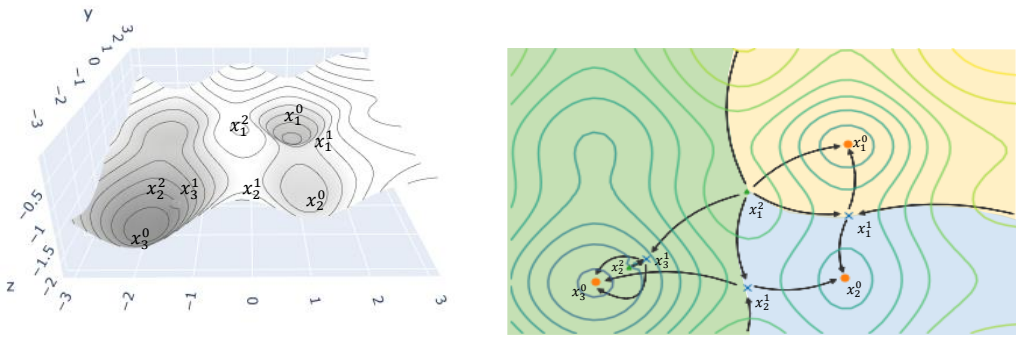
It is known that by restricting $\boldsymbol{\Sigma}_k = \sigma \boldsymbol{I}$ or all k, MoG reduces to k-means clustering when $\sigma \to \infty$.

6.2.2 Morse Theory

Let $\{\boldsymbol{x}_i\} \subset \mathcal{X}$ be a given data set of N points, with $\mathcal{X} := \mathbb{R}^D$, the data space. Given a smooth real-valued function $f : \mathcal{X} \to \mathbb{R}$ mapping each point to its height, then the inverse image of a point $a \in \mathbb{R}$ called a level set can be decomposed into several separate connected components $C_i,\ i = 1, ..., m$, i.e.,

$$\begin{aligned} \mathcal{X}_a :=& f^{-1}(-\infty, a] \\ =& \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}) \leq a\} = C_1 \cup \cdots \cup C_m \end{aligned} \tag{6.6}$$

A state vector $\boldsymbol{x}$ satisfying the equation $\nabla f(\boldsymbol{x}) = 0$ is called an *equilibrium vector* (or *critical point*) of f. We say that an equilibrium vector $\boldsymbol{x}$ of (6.8) is *hyperbolic* if the Hessian of f at $\boldsymbol{x}$ restricted to the tangent space to $\mathcal{X}$ at $\boldsymbol{x}$, denoted by $H_f(\boldsymbol{x})$, has no zero eigenvalues. Note that all the eigenvalues of the Hessian of f are real since they are real symmetric matrices. A hyperbolic equilibrium vector $\boldsymbol{x}$ is called



(a) (b)

Figure 6.1: Illustration of Morse theory. (b) is a level curve of (a). In (b), the points with marker 'o' are stable equilibrium points, and 'x' refers to index-one equilibrium point. The triangle markers refer to index-two equilibrium point. Note that $\boldsymbol{x}_3^1$ is an index-one equilibrium vector, but not a transition equilibrium vector

an *index-k equilibrium vector* if $H_f(\boldsymbol{x})$ has exactly k negative eigenvalues. The index corresponds to the dimension of the subspace consisting of directions at which f decreases. We will display the superscript k when $\boldsymbol{x}^k$ is an equilibrium vector for f having index k. We call f *separating* if distinct equilibrium vectors for f have distinct functional values.

We say that f is a *Morse function* if all the equilibrium vectors of f are hyperbolic and *separating* if distinct equilibrium vectors for f have distinct functional values. A basic result of Morse theory [82, 135] is that the class of the Morse functions forms an open, dense subset of all the smooth functions in the C^2-topology, in other words, almost all smooth functions are Morse functions. Therefore, in this paper, it is assumed that the function considered is Morse, which is "generic".

Morse theory gives the answer to the question of when the topology of $\#|\mathcal{X}_a|$ changes as a varies as follows: (Here $\#|A|$ denotes the number of connected components of a set A. See [82, 92, 135] for more details.)

- $\#|\mathcal{X}_a|$ increases by one, i.e. $\#|\mathcal{X}_{a+\varepsilon}| = \#|\mathcal{X}_{a-\varepsilon}| + 1$ for a sufficiently small $\varepsilon > 0$, if and only if, $a \in \{f(\boldsymbol{x}_1^0), ..., f(\boldsymbol{x}_s^0)\}$.

- $\#|\mathcal{X}_a|$ decreases by one, i.e. $\#|\mathcal{X}_{a+\varepsilon}| = \#|\mathcal{X}_{a-\varepsilon}| - 1$ for a sufficiently small $\epsilon > 0$, if and only if, $a \in \{f(\boldsymbol{x}_1^1), ..., f(\boldsymbol{x}_t^1)\}$ and also the following Morse relation holds.

$$\begin{aligned} H_0(\mathcal{X}_{a-\varepsilon}) &\cong H_0(\mathcal{X}_{a+\varepsilon}) \oplus \mathbb{R}, \\ H_q(\mathcal{X}_{a-\varepsilon}) &\cong H_q(\mathcal{X}_{a+\varepsilon}), \quad \text{for} \quad q > 0 \end{aligned} \tag{6.7}$$

  where $H_q(\cdot)$ is the q-th homology space and $\cong$ implies homotopy equivalent. Such $\boldsymbol{x}_i^1$ are called called a *transition equilibrium vector* (TEV) (Note in Figure 6.1, $\boldsymbol{x}_1^1$, $\boldsymbol{x}_2^1$ are TEVs, but $\boldsymbol{x}_3^1$ is not)

- $\#|\mathcal{X}_a|$ remains constant, i.e. $\#|\mathcal{X}_{a+\varepsilon}| = \#|\mathcal{X}_{a-\varepsilon}|$ for a sufficiently small $\varepsilon > 0$, if and only if, a passes the value $f(\boldsymbol{x}^k)$ of an index-k equilibrium vector $\boldsymbol{x}^k$ with $k > 1$.

6.2.3 Dynamical System Perspective

Morse theory in itself is not directly applicable due to the difficulty of computing the q-th homology space $H_q(\cdot)$, for example. The generalized gradient vector fields originated by [157] can help provide a way to compute it. Associated with the Morse function f, we can build the following generalized gradient system:

$$\frac{d\boldsymbol{x}}{dt} = -\text{grad}_R f(\boldsymbol{x}) \equiv -R(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x}). \tag{6.8}$$

where $R(\cdot)$ is a *Riemannian metric* on $\mathcal{X}$ (i.e. $R(\boldsymbol{x})$ is a positive definite symmetric matrix for all $\boldsymbol{x} \in \mathcal{X}$. The existence of a unique solution (or trajectory) $\boldsymbol{x}(\cdot) : \mathbb{R} \to \mathcal{X}$

for each initial condition $\boldsymbol{x}(0)$ is guaranteed by the smoothness of the function f [73, 99] (i.e. f is twice differentiable). Without loss of generality, we will assume that the trajectory $\boldsymbol{x}(\cdot)$ is defined on all $t \in \mathbb{R}$ for any initial condition $\boldsymbol{x}(0)$, which can be shown under a suitable re-parametrization [73]. [157] shows that system (6.8) is arbitrarily close to a C^∞-vector field and satisfies the transversality conditions on the equilibrium vectors. Also all the generalized gradient system have the equilibrium vectors at the same locations with the same index, i.e., if $R_1(\boldsymbol{x})$ and $R_2(\boldsymbol{x})$ are Riemannian metrics on $\mathbb{R}^D$, then the locations and the indices of the equilibrium vectors of $\text{grad}_{R_1} f(\boldsymbol{x})$ and $\text{grad}_{R_2} f(\boldsymbol{x})$ are the same. This property helps design computationally efficient algorithms.

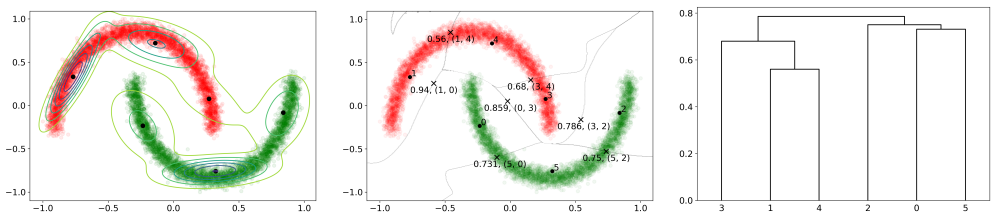
A hyperbolic equilibrium vector is called a (asymptotically) *stable* equilibrium vector (or an *attractor*), denoted by $\boldsymbol{x}^0$, if all the eigenvalues of its corresponding Jacobian are positive and an *unstable* equilibrium vector (or a *repellor*), denoted by $\boldsymbol{x}^D$, if all the eigenvalues of its corresponding Jacobian are negative. A basic result is that every local minimum of Morse function f corresponds to an (asymptotically) stable equilibrium vector of system (6.8). The (practical) *basin cell* of attraction of a stable equilibrium vector (SEV) $\boldsymbol{x}^0$ is the closure of an open and connected stable manifold, defined by

$$\mathfrak{B}(\boldsymbol{x}^0) := \text{cl}(\{\boldsymbol{x}(0) \in \mathcal{X} : \lim_{t\to\infty} \boldsymbol{x}(t) = \boldsymbol{x}^0\}).$$

where its boundary is denoted by $\partial\mathfrak{B}(\boldsymbol{x}^0)$. A basin cell groups similar data points through system (6.8).

Two SEVs, $\boldsymbol{x}_i^0$ and $\boldsymbol{x}_j^0$, are said to be *adjacent* to each other if there exists

an index-one equilibrium vector $\boldsymbol{x}_{ij}^1 \in \mathfrak{B}(\boldsymbol{x}_i^0) \cap \mathfrak{B}(\boldsymbol{x}_j^0)$. It can be shown [109, 37] that such an index-one equilibrium vector is in fact a transition equilibrium vector (TEV) between $\boldsymbol{x}_a^0$ and $\boldsymbol{x}_b^0$ that satisfies the Morse relation (6.7) and can therefore be computable by utilizing system (6.8).



(a) (b) (c)

Figure 6.2: Description of each step of the proposed method. (a) A mixture of six Gaussian distributions are obtained as a result of differentially private clustering ($\epsilon = 1$, ARI=0.359). (b) TEVs between adjacent centers are obtained, and the weight between two centers are calculated as the density of the corresponding TEV. TEVs are plotted as 'x', and the number at the bottom left of each TEV indicates its density. Two numbers in parentheses indicates which centers each TEV connects. (c) A dendrogram can be drawn according to the weighted graph constructed in (b) (ARI=1 when K=2).

6.3 Proposed Method

The proposed method consists of three steps. In the first step, differentially private parameters of the density function f, which is assumed to be a Gaussian mixture, are estimated. Then according to the Morse theory, we find TEVs of f and construct a graph that consists of the TEVs and the centers of the Gaussian mixture. Finally, we connect the adjacent centers concerning the density of the corresponding TEVs. Figure 6.2 depicts each step. In the rest of this chapter, we demonstrate each of the three steps in depth.

6.3.1 Differentially private clustering

Although our method can be applied to any Morse function, we restrict our interest to Gaussian mixture, which is

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k (2\pi)^{-\frac{D}{2}} |\mathbf{\Sigma}_k|^{-\frac{1}{2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \mathbf{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)}, \tag{6.9}$$

where $\boldsymbol{\mu}_k \in \mathbb{R}^D, \mathbf{\Sigma}_k \in \mathbb{R}^{D\times D}$ are the mean and the covariance of each normal distribution and π_k is the probability of belonging to the k-th cluster. Its associated MoG gradient system can be written as follows:

$$\frac{d\boldsymbol{x}}{dt} = R(\boldsymbol{x})^{-1} \nabla \ln p(\boldsymbol{x}) = -R(\boldsymbol{x})^{-1} \sum_{k=1}^{K} \omega_k(\boldsymbol{x}) \mathbf{\Sigma}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k), \tag{6.10}$$

where $R(\cdot)$ is a *Riemannian metric* on $\mathcal{X}$ and

$$\omega_k(\boldsymbol{x}) = \frac{\pi_k (2\pi)^{-\frac{D}{2}} |\mathbf{\Sigma}_k|^{-\frac{1}{2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \mathbf{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)}}{\sum_{k=1}^{K} \pi_k (2\pi)^{-\frac{D}{2}} |\mathbf{\Sigma}_k|^{-\frac{1}{2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \mathbf{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)}} > 0.$$

This is because the majority of the studies are concentrated on k-means clustering or mixture of Gaussians, from which the parameters of the Gaussian mixture can be naturally estimated.

Differentially private k-means clustering There have been various studies on k-means clustering algorithm that satisfy DP [15, 131, 84, 160, 154, 43, 57]. Each method is different in privacy or utility analysis, but our method can be applied to any of them because they commonly output centers and which center each sample is allocated to. Among them, DPLloyd [15] adds Laplace noise to the estimated

centers. Because each center is calculated by dividing the sum of samples belonging to the corresponding cluster by the number of samples, DPLloyd adds the corresponding noise to each of the sum and the number of samples. Because each data sample belongs to only one cluster at each iteration, the DP guarantee of DPLloyd does not change with the number of clusters, due to the parallel composition theorem. However, because k-means clustering is a kind of hard clustering, i.e., each sample is associated with a center with probability 1, it does not output the density function of the data. Therefore, we artificially build a Gaussian density function by calculating the covariance matrix of each cluster and using it as the covariance Σ_k of each Gaussian distribution. Algorithm 6 presents the algorithm for assigning a MoG density function to DPLloyd.

Algorithm 6 DPLloyd-MoG

Input: Input data $\{\boldsymbol{x}_n\}_{n=1}^N \in [-1,1]^{N\times D}$, number of cluster K, privacy budget (ϵ, δ), number of iterations τ.
Output: Parameters $\{\pi_k\}_{k=1}^K, \{\boldsymbol{\mu}_k\}_{k=1}^K, \{\boldsymbol{\Sigma}_k\}_{k=1}^K$
$\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2 ..., \boldsymbol{\mu}_K\} \leftarrow$ Randomly generate D dimensional points from uniform distribution
$r \leftarrow (2D+1)^2\tau^2 + D(2D-1)$; $\quad \sigma \leftarrow \sqrt{\frac{\tau}{2}}\frac{\sqrt{\log(1/\delta)+\epsilon}+\sqrt{\log(1/\delta)}}{\epsilon}$
while iterate until τ times **do**
  for $k = 1$ to K **do**
    Cluster $C_k \leftarrow \{\boldsymbol{x}_n : \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\| \leq \|\boldsymbol{x}_n - \boldsymbol{\mu}_i\|, \forall 1 \leq i \leq k\}$
    $N_k \leftarrow |C_k| + Lap(\sigma, size = 1)$
    $\boldsymbol{\mu}_k \leftarrow \frac{1}{N_k}(\sum_{n\in C_k} \boldsymbol{x}_n + Lap(\sigma, size = D))$
  end for
end while
for $k = 1$ to K **do**
  $\pi_k \leftarrow N_k/N$
  $\boldsymbol{\Sigma}_k^\tau \leftarrow \frac{1}{N_k}(\sum_{n\in C_k}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T + \mathsf{sym}(\mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_{D(D+1)/2})))$
end for

Differentially private mixture of Gaussians Similar to DPLloyd, [138] proposed a differentially private MoG (DPMoG) method by adding noise to parameters μ_k, Σ_k, π_k in M-step of EM algorithm. Unlike k-means, in MoG each parameter is calculated using all the samples, and thus in their method the amount of the noise added should be larger in proportion to the number of the clusters. We found that this property significantly degrades the performance of DP-MoG. Furthermore, in our method, we create a larger number of sub-clusters to represent the complex clusters, which fatally degrades the utility of DPMoG. Therefore, we present a modified version of DPMoG, whose performance is not significantly affected by the number of clusters. To enable parallel composition, we transform the responsibility obtained in E step of DPMoG so that each sample is assigned with a probability of 1 to the cluster with the largest responsibility. This technique has been studied as hard EM [97, 25, 150] in some literatures, and we are the first to apply hard EM to enhance DP. We refer to the method as DPMoG-hard.

The detailed procedure of DPMoG-hard is presented in Algorithm 7. In the algorithm, writing a distribution means random sampling from that distribution. sym is a function that transforms a $D(D+1)/2$-dimensional vector into a $D \times D$-dimensional upper-triangular matrix and then copies the matrix to be symmetric.

The sensitivity of N_k is 1, and the sensitivity of each coordinate of $\sum_{n \in C_k} \boldsymbol{x}_n$ is 2. $\sum_{n \in C_k} \boldsymbol{x}_n \boldsymbol{x}_n^T$ has $D(D+1)/2$ unique elements, among which D diagonal elements have sensitivity 1 and the other $D(D-1)/2$ elements have sensitivity 2. Using remark 2.7 and 2.8, N_k is $(1/2\sigma^2)$-zCDP, $\sum_{n \in C_k} \boldsymbol{x}_n$ (plus noise) is $(2^2 D/2\sigma^2)$-zCDP, and $\sum_{n \in C_k} \boldsymbol{x}_n \boldsymbol{x}_n^T$ (plus noise) is $(D/2\sigma^2)$-zCDP for diagonal elements and $(2^2 D(D-1)/4\sigma^2)$-zCDP for the others for an iteration. Because $\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ can be calculated

Algorithm 7 DPMoG-hard

Input: Input data $\{\boldsymbol{x}_n\}_{n=1}^N \in [-1,1]^{N\times D}$, initial parameters $\{\boldsymbol{\mu}_k^0\}_{k=1}^K, \{\boldsymbol{\Sigma}_k^0\}_{k=1}^K$, privacy budget (ϵ,δ), number of iterations τ
Output: Parameters $\{\pi_k^\tau\}_{k=1}^K, \{\boldsymbol{\mu}_k^\tau\}_{k=1}^K, \{\boldsymbol{\Sigma}_k^\tau\}_{k=1}^K$

$r \leftarrow 1+3D+2D^2$; $\sigma \leftarrow \sqrt{\frac{r\tau}{2}}\frac{\sqrt{\log(1/\delta)+\epsilon}+\sqrt{\log(1/\delta)}}{\epsilon}$
for $k=1$ to K **do**
  $C_k \leftarrow \emptyset$
end for
for $t=0$ to $\tau-1$ **do**
  //E-step
  for $n=1$ to N **do**
    for $k=1$ to K **do**
      $\gamma_{nk}^{t+1} \leftarrow \pi_k^t \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k^t,\boldsymbol{\Sigma}_k^t)/\sum_{l=1}^K \pi_l^t \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_l^t,\boldsymbol{\Sigma}_l^t)$
    end for
    $k^* = \arg\max_k \gamma_{nk}^{t+1}$; $C_{k^*} \leftarrow C_{k^*} \cup \{n\}$
  end for
  //M-step
  for $k=1$ to K **do**
    $N_k \leftarrow |C_k| + \mathcal{N}(0,\sigma^2)$; $\pi_k^{t+1} \leftarrow N_k/N$; $\boldsymbol{\mu}_k^{t+1} \leftarrow \frac{1}{N_k}(\sum_{n\in C_k}\boldsymbol{x}_n + \mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I}_D))$
    $\boldsymbol{\Sigma}_k^{t+1} \leftarrow \frac{1}{N_k}(\sum_{n\in C_k}(\boldsymbol{x}_n-\boldsymbol{\mu}_k^{t+1})(\boldsymbol{x}_n-\boldsymbol{\mu}_k^{t+1})^T + \mathsf{sym}(\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I}_{D(D+1)/2})))$
  end for
end for

with these three terms, by remark 2.8, τ iterations of the algorithm results in $(1+3D+2D^2)\tau/2\sigma^2$-zCDP. According to remark 2.5, the algorithm 7 is (ϵ,δ)-DP with $\sigma = \sqrt{\frac{r\tau}{2}}\frac{\sqrt{\log(1/\delta)+\epsilon}+\sqrt{\log(1/\delta)}}{\epsilon}$, where $r = 1+3D+2D^2$.

6.3.2 Transition equilibrium vectors and the weighted graph

To incorporate the ability of system (6.10) to generate clusters of arbitrary shapes, we need to be able to group similar basin cells based on of mutual proximity or dissimilarity. With the aid of the adjacent SEVs and TEVs, we can build the following weighted graph $G=(V,E)$ as in [92] with a derived distance:

1. The vertices V of G are the SEVs, $\boldsymbol{x}_1^0, ..., \boldsymbol{x}_s^0$, $i = 1, ..., s$ of (6.8).

2. The edge E of G can only connect vertices of adjacent SEVs, say $\boldsymbol{x}_i^0, \boldsymbol{x}_j^0$ with the edge weight, $d_E(\boldsymbol{x}_i^0, \boldsymbol{x}_j^0) := f(\boldsymbol{x}_{ij}^1)$ where $\boldsymbol{x}_{ij}^1$ is a TEV between $\boldsymbol{x}_i^0$ and $\boldsymbol{x}_j^0$.

Corresponding to $\mathcal{X}_a$, we can then build a sub-graph of G, denoted by $G_a = (V_a, E_a)$ with the following elements:

1. The vertices $V_a \subset V$ of G_a consists of SEVs, $\boldsymbol{x}_i^0$, in V with $f(\boldsymbol{x}_i^0) < a$.

2. The edge $E_a \subset E$ of G_a consists of $(\boldsymbol{x}_i^0, \boldsymbol{x}_j^0) \in E$ with $d_E(\boldsymbol{x}_i^0, \boldsymbol{x}_j^0) < a$.

The next result establishes the dynamical property of the graph G_a showing the equivalence of the topological structures between a graph G_a and the connected components of $\mathcal{X}_a$.

Proposition 6.1. *[110] With respect to the MoG (6.9), $\boldsymbol{x}_i^0$ and $\boldsymbol{x}_j^0$ are in the same connected component of the sub-graph G_a if, and only if, $\boldsymbol{x}_i^0$ and $\boldsymbol{x}_j^0$ are in the same cluster of the level set $\mathcal{X}_a$, that is, each connected component of G_a corresponds to a cluster of $\mathcal{X}_a$.*

One distinguished feature of the MoG system (6.10) is the complete stability, i.e. every trajectory converges one of the SEVs almost surely when system (6.10) is applied. Although the traditional density-based clustering methods try to assign the same label to the point in the same connected component C_i, they need retraining to assign a label to a new test point, which wastes privacy budget.

Algorithm 8 demonstrates the procedures for finding TEVs and the weighted graph G. Instead of SEPs, we use the centers $\{\boldsymbol{\mu}_k\}$ obtained in the first step, because the density of a center is very close to the local minimum, and thus there exists a SEV

nearby the center. Compared to previous studies, we can omit additional steps to find SEVs from the training samples, which requires excessive computation time. To find TEVs, we present an efficient modification of the quadratic string search method in [108], by constraining $\boldsymbol{m}^t$ in each iteration to be the vertex of the quadratic function $y = -x^2 + x$ in the transformed coordinate.

Algorithm 8 Finding Transition Equilibrium Vectors & Constructing Weighted Graph

Input: Centers $\{\boldsymbol{\mu}_k\}_{k=1}^K$, density function f, line search parameter m, number of iterations τ
Output: Weighted graph $G = (V, E)$
$V \leftarrow \{\boldsymbol{\mu}_k\}_{k=1}^K$
$TPs \leftarrow \emptyset$ *//Set of candidate transition points*
for $k = 1$ to K **do**
  for $l = k + 1$ to K **do**
    $i^* \leftarrow \arg\max_{i \in \{1,\dots,m\}} f(\mu_k + \frac{i}{m+1}(\mu_l - \mu_k))$
    $\boldsymbol{m}^0 \leftarrow \mu_k + \frac{i^*}{m+1}(\mu_l - \mu_k)$; $\boldsymbol{m}^0 \leftarrow$ Numerically integrate (6.8) from $\boldsymbol{m}^0$
    for $t = 1$ to τ **do**
      $\boldsymbol{u} \leftarrow \boldsymbol{\mu}_l - \boldsymbol{\mu}_k$; $\boldsymbol{v} \leftarrow \boldsymbol{m}^0 - \boldsymbol{\mu}_k$
      $i^* \leftarrow \arg\max_{i \in \{1,\dots,m\}} f(\mu_k + \frac{i}{m+1}\boldsymbol{u} + (\frac{i}{m+1} - (\frac{i}{m+1})^2)(4\boldsymbol{v} - 2\boldsymbol{u}))$
      $\boldsymbol{m}^t \leftarrow \mu_k + \frac{i^*}{m+1}\boldsymbol{u} + (\frac{i^*}{m+1} - (\frac{i^*}{m+1})^2)(4\boldsymbol{v} - 2\boldsymbol{u})$
      $\boldsymbol{m}^t \leftarrow$ Numerically integrate (6.8) from $\boldsymbol{m}^t$
    end for
    $\boldsymbol{t}_{tmp} \leftarrow$ Find the solution of $\nabla f(\boldsymbol{x}) = 0$ from $\boldsymbol{m}^\tau$; $TPs \leftarrow TPs \cup \{\boldsymbol{t}_{tmp}\}$
  end for
end for
for $\boldsymbol{t} \in TPs$ **do**
  if Hessian $\nabla^2 f(\boldsymbol{t})$ has one negative eigenvalue **then**
    $\boldsymbol{e} \leftarrow$ Eigenvector corresponding to the negative eigenvalue
    $\boldsymbol{x}_0 \leftarrow \boldsymbol{t} + \varepsilon\boldsymbol{e}$; $\boldsymbol{x}_1 \leftarrow \boldsymbol{t} - \varepsilon\boldsymbol{e}$ for small $\varepsilon > 0$
    $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1 \leftarrow$ Numerically integrate (6.8) from $\boldsymbol{x}_0, \boldsymbol{x}_1$
    if $\boldsymbol{\mu}_0 \neq \boldsymbol{\mu}_1$ **then**
      $E \leftarrow E \cup \langle \boldsymbol{\mu}_0, \boldsymbol{\mu}_1, f(\boldsymbol{t}) \rangle$
    end if
  end if
end for

6.3.3 Hierarchical merging of sub-clusters

The last step is a modified version of Kruscal's algorithm for minimum cost-spanning tree. Let the number of clusters, K, be arbitrarily given. The method begins with every SEV representing a singleton cluster. Denote these clusters $C_1 = \{\boldsymbol{x}_1^0\}, ..., C_L = \{\boldsymbol{x}_L^0\}$. At each of the $L-1$ steps the closest two clusters (i.e., two separate clusters containing two adjacent SEVs with the least edge weight distance) are merged into a single cluster, producing one less cluster at the next higher level. This process is terminated when we get K clusters starting from L clusters. The detailed procedure is demonstrated in Algorithm 9.

Algorithm 9 Hierarchically Merging Sub-clusters

Input: Weighted graph $G = (V, E)$, number of clusters K
Output: Clusters $\{C\}_{k=1}^{K}$
$L \leftarrow |V|$
$C_1 \leftarrow \{\boldsymbol{\mu}_1\}, \ldots, C_L \leftarrow \{\boldsymbol{\mu}_L\}$ *//Set initial clusters*
//Set initial distances
if $\langle \boldsymbol{\mu}_i, \boldsymbol{\mu}_j, f(\boldsymbol{t}) \rangle \in E$ **then**
  $d(C_i, C_j) \leftarrow f(\boldsymbol{t})$
else
  $d(C_i, C_j) \leftarrow \infty$
end if
for $l = 1$ to $L - K$ **do**
  Find the smallest d and the corresponding C_a, C_b
  $C_{L+l} \leftarrow C_a \cup C_b$; $d(C_{L+1}, C_c) = \min\{d(C_a, C_c), d(C_b, C_c)\}$ for all remaining C_cs
  Remove C_a and C_b
end for

6.4 Theoretical Results

The next result shows an inductive property of system (6.10), which provides a way to assign a label to a new test point without retraining.

Theorem 6.2 (Inductive Property)**.** *Suppose that the Riemannian metric $R(\cdot)$ on $\mathcal{X}$ of the associated MoG gradient system (6.10) has a finite condition number. Then the whole data space are almost surely pairwise disjoint union of the basin cells $\mathfrak{B}(\boldsymbol{x}_i^0)$ where $\boldsymbol{x}_i^0, i = 1, ..., s$ are the SEVs of system (6.10), i.e.*

$$\mathcal{X} = \mathfrak{B}(\boldsymbol{x}_1^0)\dot{\cup}\cdots\dot{\cup}\mathfrak{B}(\boldsymbol{x}_s^0)$$

Here almost surely disjoint union $A\dot{\cup}B$ of two nonempty sets A and B means that $A \cap B$ has a Lebesgue measure zero.

Proof. Following the proof of Lasalle's invariance property theorem as in [73, 98], it can be easily shown that every bounded trajectory of system (6.10) converges to one of the equilibrium vectors. Therefore it is enough to show that every trajectory is bounded.

Since $R(\boldsymbol{x})^{-1}$ and Σ_k^{-1} are positive definite, we can let $A_k(\boldsymbol{x})$ be the Cholesky factorization of the positive definite matrix $R(\boldsymbol{x})^{-1}\Sigma_k^{-1}$ that satisfies $R(\boldsymbol{x})^{-1}\Sigma_k^{-1} = A_k(\boldsymbol{x})^T A_k(\boldsymbol{x})$. Then $(\boldsymbol{x}^T R(\boldsymbol{x})^{-1}\Sigma_k^{-1}\boldsymbol{x}) = \|A_k(\boldsymbol{x})\boldsymbol{x}\|^2$. Since the condition number of $R(\boldsymbol{x})$ is bounded, by the spectral theorem, there exist positive smooth eigenvalue functions $\lambda_{\min}^k(\boldsymbol{x}), \lambda_{\max}^k(\boldsymbol{x}) > 0$ and $\gamma > 0$ such that

$$\|A_k(\boldsymbol{x})\| = \sqrt{\lambda_{\max}^k(\boldsymbol{x})}, \quad \forall k = 1, ...K, \ \forall \boldsymbol{x} \in \mathcal{X},$$
$$\kappa(A_k(\boldsymbol{x})) := \left(\lambda_{\max}^k(\boldsymbol{x})/\lambda_{\min}^k(\boldsymbol{x})\right)^{1/2} \leq \gamma, \quad \forall k = 1, ...K, \ \forall \boldsymbol{x} \in \mathcal{X}$$

where $\|\cdot\|$ denotes the Euclidean norm or ℓ_2-norm. Now let $V(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|^2$ and choose $\Upsilon > \gamma \max_k \|\boldsymbol{\mu}_k\|$. Then for any $L > \Upsilon$, and for all $\|\boldsymbol{x}\| = L$, we have

$$\begin{aligned}
\frac{\partial}{\partial t}V(\boldsymbol{x}) &= \boldsymbol{x}^T\frac{d\boldsymbol{x}}{dt} = -\boldsymbol{x}^T\sum_k \omega_k(\boldsymbol{x})R(\boldsymbol{x})^{-1}\Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) \\
&= -\boldsymbol{x}^T\sum_k \omega_k(\boldsymbol{x})R(\boldsymbol{x})^{-1}\Sigma_k^{-1}\boldsymbol{x} + \boldsymbol{x}^T\sum_k \omega_k(\boldsymbol{x})R(\boldsymbol{x})^{-1}\Sigma_k^{-1}\boldsymbol{\mu}_k \\
&= -\sum_k \omega_k(\boldsymbol{x})\boldsymbol{x}^T A_k(\boldsymbol{x})^T A_k(\boldsymbol{x})\boldsymbol{x} + \sum_k \omega_k(\boldsymbol{x})\boldsymbol{x}^T A_k(\boldsymbol{x})^T A_k(\boldsymbol{x})\boldsymbol{\mu}_k \\
&\leq -\sum_k \omega_k(\boldsymbol{x})\|A_k(\boldsymbol{x})\boldsymbol{x}\|^2 + \sum_k \omega_k(\boldsymbol{x})\|A_k(\boldsymbol{x})\boldsymbol{x}\|\|A_k(\boldsymbol{x})\boldsymbol{\mu}_k\| \\
&= \sum_k \omega_k(\boldsymbol{x})\|A_k(\boldsymbol{x})\boldsymbol{x}\|(\|A_k(\boldsymbol{x})\boldsymbol{\mu}_k\| - \|A_k(\boldsymbol{x})\boldsymbol{x}\|) \\
&\leq \sum_k \omega_k(\boldsymbol{x})\|A_k(\boldsymbol{x})\boldsymbol{x}\|(\sqrt{\lambda^k_{\max}(\boldsymbol{x})}\|\boldsymbol{\mu}_k\| - \sqrt{\lambda^k_{\min}(\boldsymbol{x})}\|\boldsymbol{x}\|) \\
&< 0
\end{aligned}$$

Therefore, for any $L > \Upsilon$, the trajectory starting from any point on $\|\boldsymbol{x}\| = L > \Upsilon$ always enters into the bounded set $\|\boldsymbol{x}\| \leq L$, which implies that $\{\boldsymbol{x}(t) : t \geq 0\}$ is bounded. □

This result naturally partitions the sample space by assigning points belonging to different basin cells to their respective SEVs. Moreover, it is computationally feasible to identify a basin cell of a SEV by applying the system (6.10).

The next result shows the dynamical invariance property of the MoG system (6.10) that preserves differential privacy. The result implies that Algorithm 8 does not incur any additional privacy loss to the differentially private clustering algorithms.

Theorem 6.3 (Preserving Differential Privacy)**.** *Let* $\mathcal{M}$ *with* Range$(\mathcal{M}) \subseteq \mathcal{X}$ *be*

a randomized algorithm that is (ϵ, δ)-differentially private. Then under the same condition of Theorem 6.2, the inductive dynamical processing (6.10) applied to $\mathcal{M}$ is (ϵ, δ)-differentially private, i.e. the solution trajectory $\boldsymbol{x}(t)$ of the MoG gradient system (6.10) with initial condition $\boldsymbol{x} \in \mathcal{M}$ is (ϵ, δ)-differentially private.

Proof. Define the flow $\Phi_t : \mathcal{X} \to \mathcal{X}$ of the MoG gradient system (6.10) by $\Phi_t(\boldsymbol{x}) = \boldsymbol{x}(t)$ with initial condition $\boldsymbol{x}(0) = \boldsymbol{x}$ for $t \in \mathbb{R}$. Then by the fundamental theorem of the flow defined by system (6.10), we have $\Phi_{s+t}(\boldsymbol{x}) = \Phi_s \circ \Phi_t(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$ and $s, t \in \mathbb{R}$. (See [162, 98] for more details.) Now let a pair of neighboring databases D_1, D_2 with $\|D_1 - D_2\|_1 \leq 1$ be given. Then for all $\boldsymbol{x} \in \mathcal{M}(D_1)$

$$\boldsymbol{x}(t) = \Phi_t \circ \Phi_0(\boldsymbol{x}) = \Phi_t \circ \boldsymbol{x}(0)$$

For any event $\mathcal{O} \subset \text{Range}(\mathcal{M}) \subseteq \mathcal{X}$ and any $t \in \mathbb{R}$, we let $\mathcal{W}_t = \{\boldsymbol{x} \in \mathcal{X} : \boldsymbol{x}(t) \in \mathcal{O}\}$. Then we have

$$\begin{aligned} \Pr\{\boldsymbol{x}(t) \in \mathcal{O} : \boldsymbol{x} \in \mathcal{M}(D_1)\} &= \Pr[\mathcal{M}(D_1) \in \mathcal{W}_t] \\ &\leq e^{\epsilon} \Pr[\mathcal{M}(D_2) \in \mathcal{W}_t] + \delta \\ &= e^{\epsilon} \Pr\{\boldsymbol{x}(t) \in \mathcal{O} : \boldsymbol{x} \in \mathcal{M}(D_2)\} + \delta \end{aligned}$$

Since this result works for any $t \in \mathbb{R}$, the inductive dynamical processing applied to $\mathcal{M}$ by system (6.10) is (ϵ, δ)-differentially private. □

Another distinguished feature of the MoG system (6.10) is the agglomerative property, i.e. system (6.10) enables us to build a hierarchy of clusters starting from the basin cells. This ensures that Algorithm 9 can obtain any desired number of

clusters.

Theorem 6.4 (Agglomerative Property)**.** *Let* $\boldsymbol{x}_i^0, i = 1, ..., s$ *and* $\boldsymbol{x}_j^1, j = 1, ..., t$ *be the SEVs and the TEVs of the MoG system (6.10), respectively. Consider an agglomerative process such that each basin cell* $\mathfrak{B}(\boldsymbol{x}_1^0)$ *starts in its own cluster and a pair of clusters are merged when the two separate clusters contain adjacent basin cells with the least edge weight* $f(\boldsymbol{x}_j^1)$*. Then the merging occurs when we decrease the level value* a *starting from* $\max_i f(\boldsymbol{x}_i^0)$ *until it hits the value in* $\{f(\boldsymbol{x}_1^1), ..., f(\boldsymbol{x}_t^1)\}$ *and this process is terminated when we get one cluster, say* $\mathcal{X}$*, starting from* s *clusters.*

Proof. The first part of the proof comes from the Morse theory. For the second part of the proof, it is sufficient to show that $\eta > 0$ exists, with $\mathcal{X}_r$ connected for all $0 < r < \eta$. From the Cholesky factorization of Σ_k^{-1}, we can let $\Sigma_k^{-1} = U_k^T U_k$ where U_k is an upper triangle matrix with positive diagonal elements. By the singular value decomposition theorem, $0 < \sigma_d^{(k)} \|\boldsymbol{x}\| \leq \|U_k \boldsymbol{x}\| \leq \sigma_1^{(k)} \|\boldsymbol{x}\|$ where $\sigma_1^{(k)} \geq \cdots \geq \sigma_d^{(k)}$ are the singular values for $k = 1, ..., K$. Let $a := \min_k \sigma_d^{(k)}$, $b := \max_k \sigma_1^{(k)}$. Choose $\zeta > \max\{\frac{b}{a}, \gamma\} \cdot \max_k \|\boldsymbol{\mu}_k\|$, then for all $\|\boldsymbol{x}\| = L > \zeta$, we have

$$\|U_k(\boldsymbol{x} - \boldsymbol{\mu}_k)\| \leq \|U_k \boldsymbol{x}\| + \|U_k \boldsymbol{\mu}_k\| < bL + \frac{a\sigma_1^{(k)}}{b} L \leq (b + a)L,$$

By the proof of Theorem 6.2, every trajectory starting from $\|\boldsymbol{x}\| = L$ for any $L > \zeta$ always enters into the $\mathcal{S}_L := \{\boldsymbol{x} : \|\boldsymbol{x}\| \leq L\}$, which is a connected and bounded set.

It is enough to show that there exist a $\eta > 0$ such that $\mathcal{S}_L \subset \mathcal{X}_\eta$.

$$\begin{aligned} p(\boldsymbol{x}) &= \sum_{k=1}^{K} \pi_k (2\pi)^{-\frac{d}{2}} |\boldsymbol{\Sigma}_k|^{-\frac{1}{2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)} \\ &> \sum_{k=1}^{K} \pi_k (2\pi)^{-\frac{d}{2}} |U_k| e^{-\frac{1}{2}(a+b)^2 L^2} \\ &\geq \min_k |U_k| (2\pi)^{-\frac{d}{2}} e^{-\frac{1}{2}(a+b)^2 L^2} \\ &\geq (2\pi/a^2)^{-\frac{d}{2}} e^{-\frac{1}{2}(a+b)^2 L^2}. \end{aligned}$$

The second inequality follows from $\sum_{k=1}^{K} \pi_k = 1$, and the last inequality follows from

$$|U_k| = \prod_{i=1}^{d} \sigma_i^{(k)} \geq (\sigma_d^{(k)})^d \geq a^d$$

When we choose $\eta = (2\pi/a^2)^{-\frac{d}{2}} e^{-\frac{1}{2}(a+b)^2 L^2}$, we have $\mathcal{S}_L \subset \mathcal{X}_r$ for all $0 < r < \eta$. Hence, by theorem 6.2, for any point $\boldsymbol{x}_0 \in (\mathcal{X}_r \setminus \mathcal{S}_L)$, every trajectory starting from $\boldsymbol{x}_0$ should hit the boundary L and enters into the region $\mathcal{S}_L$. This implies that the set $\mathcal{S}_L$ is a strong deformation retract of the level set $\mathcal{X}_r$, which shows that $\mathcal{X}_r$ is connected for all $r < \gamma$. □

6.5 Experiments

We evaluate the proposed method on various real-world datasets. Through experiments, we verify that our method achieves better clustering results compared to the existing methods.

6.5.1 Experimental Setting

Datasets We used six datasets: **Shuttle** dataset, EMG physical action (**EMG**) dataset, MAGIC gamma telescope (**MAGIC**) dataset from UCI machine learing repository [49], Sloan digital sky survey (**Sloan**)[1] dataset and Predicting pulsar star (**Pulsar**)[2] dataset from Kaggle competition, and **USPS** [86] dataset. For all datasets, we rescaled each variable in $[-1, 1]$ to ensure DP. **EMG** datset originally consists of 10 classes, but for convenience, similar actions were remove. For **USPS** dataset, we reduced the dimension from 16×16 to 3 with t-SNE, because the utility of base methods decreases as the dimension increases.

Table 6.1 contains a detailed description of the datasets used for the experiments. The lost column in the table, K_0, denotes the nubmer of sub-clusters generated in the first step of the proposed method.

Experimental Setup To compare the utility of differentially private clustering methods before and after applying Morse theory, we compared the following four methods in pairs: **DPLloyd** and **DPLloyd-Morse**, **DPMoG-hard** and **DPMoG-hard-Morse**. Here "-Morse" denotes that Morse theory was applied. For clustering metric, we used adjusted Rand index (ARI). ARI has a value between 0 and 1,

[1] https://www.kaggle.com/lucidlenn/sloan-digital-sky-survey
[2] https://www.kaggle.com/colearninglounge/predicting-pulsar-starintermediate

Table 6.1: Description of datasets.

Dataset	# of samples (N)	# of variables (D)	# of clusters (K)	K_0
Shuttle	43500	9	7	12
EMG	59130	8	6	10
MAGIC	19020	10	2	6
Sloan	10000	16	3	6
Pulsar	9273	8	2	6
USPS	7291	3	10	13

and the closer the value is to 1, the better the clustering is. Although true labels are needed to compute ARI, we found out existing clustering metrics that do not require true labels, such as silhouette score, do not increase by capturing complex shape of clusters because those metrics are suitable to fit convex cluster shapes.

Total privacy budget ϵ was set in $\{10,\ 5,\ 2,\ 1\}$. For $\epsilon < 1$, **DPLloyd** and **DPMoG-hard** showed meaningless ARI scores for most of the datasets. This result is different from the experiments using other metrics such as the k-means objective, in which performance is preserved for smaller ϵ. Also, the level of privacy can be further enhanced by using more advanced models. In addition to the privacy budget ϵ, there are several parameters to be determined. The number of iterations τ_1 and τ_2 were set 10 and 5, respectively, for all experiments. For Algorithm 8, we set the line search parameter $m = 20$, and the small perturbation for finding TEVs ε were set to 0.05. For **DPLloyd** and **DPMoG-hard** we set the number of clusters to K, and for **DPLloyd-Morse** and **DPMoG-hard-Morse** we first set the number of sub-clusters to K_0 and merged them to K clusters. The experiments were performed with Python 3.6.9, and all experiments were repeated 5 times and the average of the results was obtained.

6.5.2 Experimental Results

Effectiveness of Morse theory Figure 6.3 demonstrates the clustering results for the six datasets for the mixture of Gaussians. As shown in the figure, by applying Morse theory **DPMoG-hard-Morse** shows higher ARI than **DPMoG-hard** in all cases. Specifically, for **Pulsar** dataset, ARI increased by up to 0.2, which means a significant improvement in the utility. For other datasets, ARI generally increased by 0.05 to 0.1. The results imply that the proposed method can effectively group the generated sub-clusters to express the clusters of arbitrary shapes.

Meanwhile, for most data, the difference between ARI scores of **DPMoG-hard** and **DPMoG-hard-Morse** tends to decrease as epsilon gets smaller. Also, for **EMG** and **Sloan** datasets, the increase was relatively small for all ϵ. Both datasets in common show poor performance compared to the non-private model even at $\epsilon = 10$. Combining these facts, it is a limitation of the proposed method that applying Morse theory does not have a significant effect when the performance of the baseline model is completely degraded because the noise to guarantee DP is too large. However, **DPMoG-hard** is a relatively simple model, and this limitation can be overcome by using a better baseline model.

Table 6.2 demonstrates the results of t-test between the ARI scores of **DPMoG-hard** and **DPMoG-hard-Morse**. The results are consistent with Figure 6.3, and the significance of the difference tends to decrease as ϵ decreases. The significance of the difference increases when $\epsilon = 1$, and we interpret this result that the randomness becomes too large, resulting in unpredictable results.

Figure 6.4 demonstrates the k-means clustering results for six same datasets. The ARI score of **DPLloyd-Morse** is higher than **DPLloyd** in almost all case.

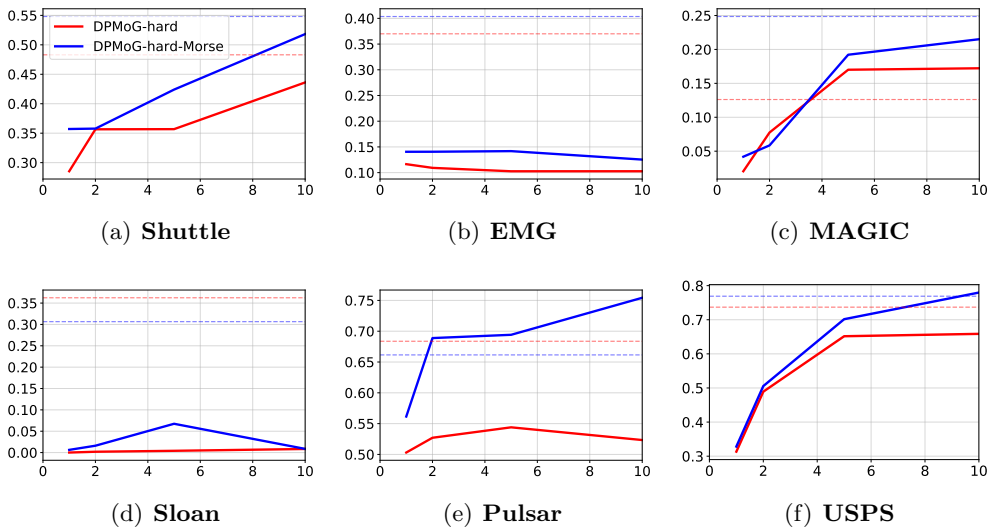
Table 6.2: T-test results between the ARI scores of **DPMoG-hard-Morse** and **DPMoG-hard.**

Dataset	$\epsilon = 10$		$\epsilon = 5$		$\epsilon = 2$		$\epsilon = 1$	
	t-statistic	**p value**	**t-statistic**	**p value**	**t-statistic**	**p value**	**t-statistic**	**p value**
Shuttle	0.387	0.354	1.219	0.125	1.377	0.099	0.961	0.180
EMG	6.454	<0.001	5.275	<0.001	-1.179	0.869	1.078	0.153
MAGIC	1.609	0.066	3.542	0.002	3.453	0.003	6.362	<0.001
Sloan	6.747	<0.001	2.930	0.006	-2.469	0.987	4.169	<0.001
Pulsar	7.847	<0.001	5.143	<0.001	7.759	<0.001	0.567	0.292
USPS	4.805	<0.001	9.673	<0.001	0.068	0.474	6.066	<0.001

In particular, the ARI score of **DPLloyd-Morse** is more than 0.2 higher than that of **DPLloyd**, which means a substantial improvement. As shown in the figure, non-private **DPLloyd-Morse** performs significantly better than **DPLloyd** in the **Magic** dataset which perform poorly in k-means algorithm. This shows that Morse theory can improve the utility of k-means algorithm. The difference between ARI scores tends to decrease as epsilon gets smaller, similar to the results for **DPMoG-hard**.

We additionally evaluated the proposed method on a differentially private k-means clustering method in [10], which is a relatively recent study. We denote their method **DPCube** because their method iteratively generates small cubes. Figure 6.5 demonstrates the results on **DPCube** and **DPCube-Morse**. It is shown that **DPCube-Morse** always shows higher clustering performance than **DPCube** in low-privacy regions. For **EMG**, **DPCube** was better when $\epsilon \leq 2$. However, the difference is not significant because in the case of **EMG**, the scale of the y-axis is very small.

Effect of increasing sub-clusters Theoretically, the proposed method can reach the desired number of clusters by grouping them no matter how many Gaussian



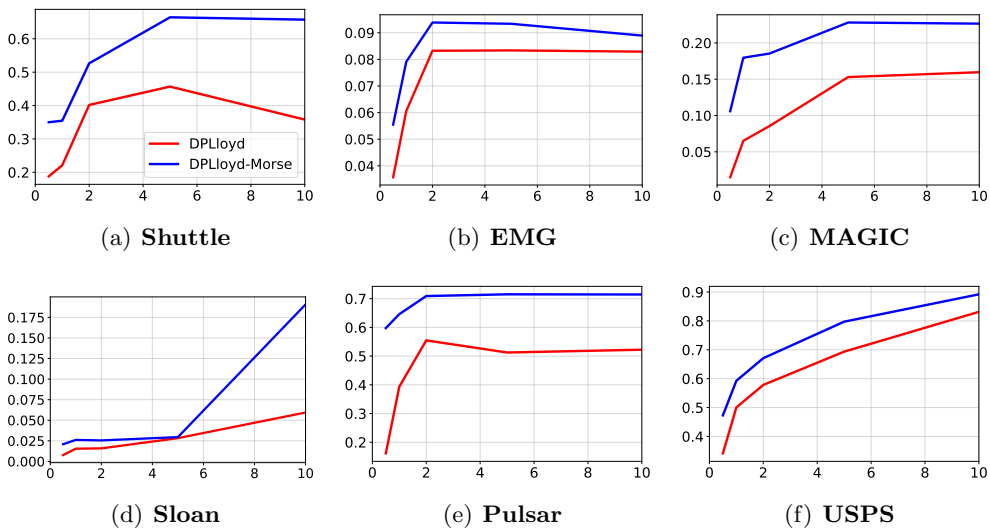
(a) **Shuttle** (b) **EMG** (c) **MAGIC**

(d) **Sloan** (e) **Pulsar** (f) **USPS**

Figure 6.3: Clustering results for real-world datasets. The x-axis indicates the noise budget ϵ, and the y-axis indicates the ARI score. The dotted lines indicate the performances of the non-private models.

sub-clusters are created. However, because different TEVs are generated with a different number of centers, the final clustering results are not the same. Therefore, we empirically measured how consistent the final clustering performance was for different numbers of sub-clusters.

Table 6.3 shows the clustering results with different numbers of initial sub-clusters, for **Pulsar** dataset. The clustering performance tends to decrease as K_0 increases. The result can be interpreted in two ways. First, the difference in performance can be caused by the increase in the number of TEVs. Another interpretation is that the increase in the number of sub-clusters reduces the number of samples belonging to each cluster, and thus the influence of noise increases.



(a) **Shuttle** (b) **EMG** (c) **MAGIC**

(d) **Sloan** (e) **Pulsar** (f) **USPS**

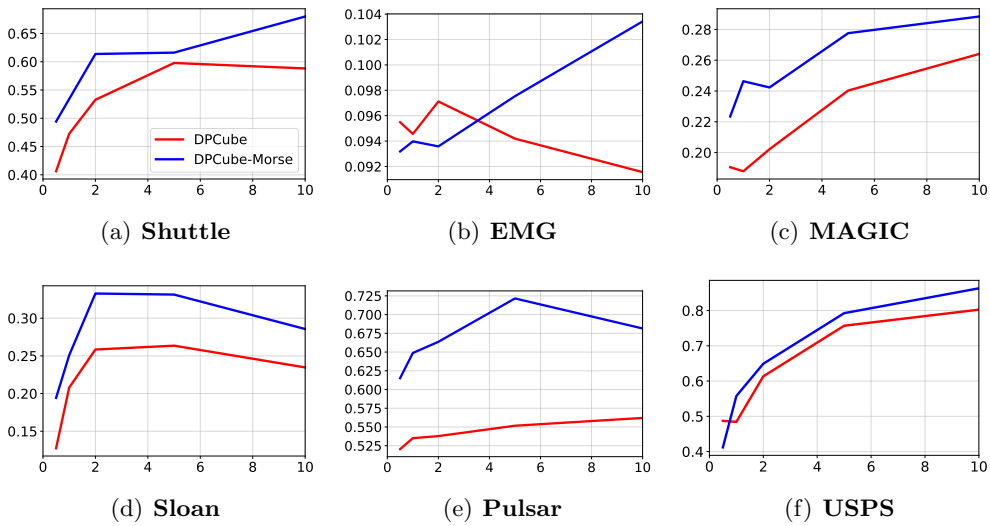
Figure 6.4: Clustering results for real-world datasets. The x-axis indicates the noise budget ϵ, and the y-axis indicates the ARI score. The red line shows the performance of **DPLloyd**, and the blue line shows the performance of **DPLloyd-Morse**.

Table 6.3: ARI scores with different numbers of initial sub-clusters for **Pulsar** dataset.

ϵ \ K_0	**6**	**10**	**15**	**20**
10	0.754277	0.754838	0.766507	0.728404
5	0.694178	0.673453	0.659280	0.664384
2	0.688862	0.610148	0.596477	0.576486
1	0.561494	0.584608	0.577082	0.542264

6.6 Chapter Summary

We proposed an effective differentially private clustering method, which utilizes Morse theory to express complex, nonconvex clusters without sacrificing privacy. The proposed method first generates many convex sub-clusters using mixture of Gaussians or k-means clustering, and hierarchically connects the sub-clusters using a hierarchical procedure based on Morse theory. As theoretical results, we proved the dynamical processing associated with mixture of Gaussians is completely stable,



(a) **Shuttle** (b) **EMG** (c) **MAGIC**

(d) **Sloan** (e) **Pulsar** (f) **USPS**

Figure 6.5: Clustering results for real-world datasets. The x-axis indicates the noise budget ϵ, and the y-axis indicates the ARI score. The red line shows the performance of **DPCube**, and the blue line shows the performance of **DPCube-Morse**.

and does not add any privacy loss on the existing methods. The proposed method showed better clustering utility for various real-world datasets. How the number of sub-clusters affects the performance of the proposed method should be further studied.

Chapter 7

Conclusion

7.1 Conclusion

As machine learning has achieved successful results in various applications, it has been noticed that machine learning in the real world can pose threats that can expose sensitive information such as data or model information to unwanted opponents. Privacy problems arise because data ownership, computational capabilities, and the use of machine learning models are not all done by one entity. When the learning or inference process of machine learning is collaboratively performed by multiple participants, each participant may attempt to steal sensitive information owned by different participants. Accordingly, new machine learning methodologies capable of protecting sensitive information are being developed. While effectively protecting privacy, privacy-preserving machine learning methods are inferior to conventional machine learning in several ways, such as computational efficiency and model performance.

In this dissertation, we aimed to develop new privacy-preserving machine learning methods which enhances the trade-off between privacy and utility. To ensure privacy, we used homomorphic encryption and differential privacy, well-known and complementary methods. Contributions of this thesis are as follows:

1. We proposed an efficient training of ridge regression with homomorphic encryption. By encrypting only sensitive attributes, our method can avoid complex computations for matrix inversion and reduce the computation time in proportion to the number of the attributes. By using homomorphic encryption in the training phase, our method can protect privacy of the training data against the computation server and privacy of the trained model.

2. We extended the homomorpic-encryption-friendly ridge regression to develop a novel logistic regression method which is free from hyper-parameter selection. Compared to existing methods, our method does not require gradient descent and can acquire the exact solution. We additionally proposed a mean matching method to reduce the discrepancy between the distributions of protected and unprotected dataset.

3. We proposed an efficient evaluation of support vector clustering with homomorphic encryption. Compared to previous clustering methods, our method can cluster complex, non-convex datasets better. By using homomorphic encryption in the inference phase, our method can protect privacy of the test data against the computation server.

4. We introduced Morse theory into differentially private mixture of Gaussians clustering. By merging initial sub-clusters according to Morse theory, our method can express more complex cluster shapes compared to existing methods. By using differential privacy, our method can protect privacy of the training data against the external users.

7.2 Future Direction

As interest in the privacy of machine learning is increasing, privacy-preserving machine learning still has room for further development. Concluding the dissertation, we discuss some possible directions of future works that can further advance our research to protect the privacy of more diverse machine learning models or make algorithms more efficient.

1. We proposed an efficient homomorphic encryption framework for ridge regression and logistic regression by encrypting only information that needs higher level of privacy. However, it is not trivial to apply the same framework to more complex machine learning methods. For example, even if some sensitive variables are encrypted in for training a kernel support vector machine, the kernel matrix is calculated so that all of its elements are encrypted. Extending our framework to other more complex and high-use models such as kernel support vector machines and deep learning is an important research direction.

2. Our differentially private clustering method using Morse theory proposed in Chapter 6 has a limitation that it can be applied only to models with a MoG density function. However, in practice the same methodology is always applicable if the density function does not directly involve information of training data. Thus, our method can be further extended by finding more appropriate density functions. Furthermore, our method can be applied naturally to supervised learning and semi-supervised learning, in addition to clustering.

3. Differential privacy allows to effectively cope with various threats to training data against the model users. However, how resistant differential privacy is to

model extraction attacks has not yet been properly studied. Model extraction attacks are important not only because the target model itself has value, but also because the extracted model can be used for other secondary attacks. Therefore, developing an effective homomorphic-encryption-friendly defense method against model extraction attacks build an integrated system that can defend against various kinds of attacks.

Bibliography

[1] M. Abadi, 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, pp. 308–318.

[2] M. S. Acharya, A. Armaan, and A. S. Antony, *A comparison of regression models for prediction of graduate admissions*, in 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), IEEE, 2019, pp. 1–5.

[3] M. Al-Rubaie and J. M. Chang, *Privacy-preserving machine learning: Threats and solutions*, IEEE Security & Privacy, 17 (2019), pp. 49–58.

[4] N. Almutairi, F. Coenen, and K. Dures, *K-means clustering using homomorphic encryption and an updatable distance matrix: secure third party data clustering with limited data owner interaction*, in International Conference on Big Data Analytics and Knowledge Discovery, Springer, 2017, pp. 274–285.

[5] J. Alperin-Sheriff and C. Peikert, *Practical bootstrapping in quasilinear time*, in Annual Cryptology Conference, Springer, 2013, pp. 1–20.

[6] A. Amuthan and R. Sendhil, *Hybrid gsw and dm based fully homomorphic encryption scheme for handling false data injection attacks under privacy*

preserving data aggregation in fog computing, Journal of Ambient Intelligence and Humanized Computing, 11 (2020), pp. 5217–5231.

[7] Y. Aono, T. Hayashi, L. T. Phong, and L. Wang, *Privacy-preserving logistic regression with distributed data sources via homomorphic encryption*, IEICE TRANSACTIONS on Information and Systems, 99 (2016), pp. 2079–2089.

[8] ——, *Input and output privacy-preserving linear regression*, IEICE TRANSACTIONS on Information and Systems, 100 (2017), pp. 2339–2347.

[9] Y. Aono, T. Hayashi, L. Trieu Phong, and L. Wang, *Scalable and secure logistic regression via homomorphic encryption*, in Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, 2016, pp. 142–144.

[10] M.-F. Balcan, T. Dick, Y. Liang, W. Mou, and H. Zhang, *Differentially private clustering in high-dimensional euclidean spaces*, in International Conference on Machine Learning, PMLR, 2017, pp. 322–331.

[11] A. Barnett, J. Santokhi, M. Simpson, N. P. Smart, C. Stainton-Bygrave, S. Vivek, and A. Waller, *Image classification using non-linear support vector machines on encrypted data*, Cryptology ePrint Archive, (2017).

[12] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, *A theory of learning from different domains*, Machine learning, 79 (2010), pp. 151–175.

[13] A. Ben-Israel, *An iterative method for computing the generalized inverse of an arbitrary matrix*, Mathematics of Computation, (1965), pp. 452–455.

[14] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*, vol. 6, Athena Scientific Belmont, MA, 1997.

[15] A. Blum, C. Dwork, F. McSherry, and K. Nissim, *Practical privacy: the sulq framework*, in Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2005, pp. 128–138.

[16] C. Boura, N. Gama, M. Georgieva, and D. Jetchev, *Chimera: Combining ring-lwe-based fully homomorphic encryption schemes*, Journal of Mathematical Cryptology, 14 (2020), pp. 316–338.

[17] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, *Fast homomorphic evaluation of deep discretized neural networks*, in Annual International Cryptology Conference, Springer, 2018, pp. 483–512.

[18] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, *(leveled) fully homomorphic encryption without bootstrapping*, ACM Transactions on Computation Theory (TOCT), 6 (2014), pp. 1–36.

[19] Z. Brakerski and V. Vaikuntanathan, *Efficient fully homomorphic encryption from (standard) lwe*, SIAM Journal on Computing, 43 (2014), pp. 831–871.

[20] A. Brutzkus, R. Gilad-Bachrach, and O. Elisha, *Low latency privacy preserving inference*, in International Conference on Machine Learning, PMLR, 2019, pp. 812–821.

[21] M. Bun and T. Steinke, *Concentrated differential privacy: Simplifications, extensions, and lower bounds*, in Theory of Cryptography Conference, Springer, 2016, pp. 635–658.

[22] J. Byun, J. Lee, and S. Park, *Privacy-preserving evaluation for support vector clustering*, Electronics Letters, (2021).

[23] J. Byun, W. Lee, and J. Lee, *Parameter-free he-friendly logistic regression*, Advances in Neural Information Processing Systems, 34 (2021).

[24] Z. Cai, Z. He, X. Guan, and Y. Li, *Collective data-sanitization for preventing sensitive information inference attacks in social networks*, IEEE Transactions on Dependable and Secure Computing, 15 (2016), pp. 577–590.

[25] G. Celeux and G. Govaert, *A classification em algorithm for clustering and two stochastic versions*, Computational statistics & Data analysis, 14 (1992), pp. 315–332.

[26] A. Chaabane, G. Acs, M. A. Kaafar, et al., *You are what you like! information leakage through users' interests*, in Proceedings of the 19th annual network & distributed system security symposium (NDSS), Citeseer, 2012.

[27] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, *Differentially private empirical risk minimization.*, Journal of Machine Learning Research, 12 (2011).

[28] H. Chen, I. Chillotti, and Y. Song, *Improved bootstrapping for approximate homomorphic encryption*, in Advances in Cryptology – EUROCRYPT

2019, Y. Ishai and V. Rijmen, eds., Cham, 2019, Springer International Publishing, pp. 34–54.

[29] H. Chen, R. Gilad-Bachrach, K. Han, Z. Huang, A. Jalali, K. Laine, and K. Lauter, *Logistic regression over encrypted data from fully homomorphic encryption*, BMC medical genomics, 11 (2018), pp. 3–12.

[30] Y.-R. Chen, A. Rezapour, and W.-G. Tzeng, *Privacy-preserving ridge regression on distributed data*, Information Sciences, 451 (2018), pp. 34–49.

[31] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, *Bootstrapping for approximate homomorphic encryption*, in Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2018, pp. 360–384.

[32] J. H. Cheon, S. Hong, and D. Kim, *Remark on the security of ckks scheme in practice.*, IACR Cryptol. ePrint Arch., 2020 (2020), p. 1581.

[33] J. H. Cheon, A. Kim, M. Kim, and Y. Song, *Homomorphic encryption for arithmetic of approximate numbers*, in International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2017, pp. 409–437.

[34] J. H. Cheon, A. Kim, and D. Yhee, *Multi-dimensional packing for heaan for approximate matrix arithmetics.*, IACR Cryptol. ePrint Arch., 2018 (2018), p. 1245.

[35] J. H. Cheon, D. Kim, D. Kim, H. H. Lee, and K. Lee, *Numerical method for comparison on homomorphically encrypted numbers*, in International Con-

ference on the Theory and Application of Cryptology and Information Security, Springer, 2019, pp. 415–445.

[36] J. H. Cheon, D. Kim, and J. H. Park, *Towards a practical cluster analysis over encrypted data*, in International Conference on Selected Areas in Cryptography, Springer, 2019, pp. 227–249.

[37] H.-D. Chiang and L. Fekih-Ahmed, *Quasi-stability regions of nonlinear dynamical systems: Theory*, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 43 (1996), pp. 627–635.

[38] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, *Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe*, in International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2017, pp. 377–408.

[39] ——, *Tfhe: fast fully homomorphic encryption over the torus*, Journal of Cryptology, 33 (2020), pp. 34–91.

[40] E. Christodoulou, J. Ma, G. S. Collins, E. W. Steyerberg, J. Y. Verbakel, and B. Van Calster, *A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models*, Journal of clinical epidemiology, 110 (2019), pp. 12–22.

[41] K.-S. Chuang, H.-L. Tzeng, S. Chen, J. Wu, and T.-J. Chen, *Fuzzy c-means clustering with spatial information for image segmentation*, computerized medical imaging and graphics, 30 (2006), pp. 9–15.

[42] M. d. Cock, R. Dowsley, A. C. Nascimento, and S. C. Newman, *Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data*, in Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security, 2015, pp. 3–14.

[43] E. Cohen, H. Kaplan, Y. Mansour, U. Stemmer, and E. Tsfadia, *Differentially-private clustering of easy instances*, in International Conference on Machine Learning, PMLR, 2021, pp. 2049–2059.

[44] K. Copil, M. Wörbelauer, H. Meyr, and H. Tempelmeier, *Simultaneous lotsizing and scheduling problems: a classification and review of models*, OR spectrum, 39 (2017), pp. 1–64.

[45] J. L. Crawford, C. Gentry, S. Halevi, D. Platt, and V. Shoup, *Doing real work with fhe: the case of logistic regression*, in Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, 2018, pp. 1–12.

[46] S. Dalal and W. Hall, *Approximating priors by mixtures of natural conjugate priors*, Journal of the Royal Statistical Society: Series B (Methodological), 45 (1983), pp. 278–286.

[47] A. De Caigny, K. Coussement, and K. W. De Bock, *A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees*, European Journal of Operational Research, 269 (2018), pp. 760–772.

[48] J. De Spiegeleer, D. B. Madan, S. Reyners, and W. Schoutens, *Machine learning for quantitative finance: fast derivative pricing, hedging and fitting*, Quantitative Finance, 18 (2018), pp. 1635–1643.

[49] D. Dua and C. Graff, *UCI machine learning repository*, 2017.

[50] A. J. Duarte and J. Carvalho, *Discrete lot sizing and scheduling on parallel machines: description of a column generation approach*, in IO2013-XVI Congresso da Associação Portuguesa de Investigação Operacional, 2013, pp. 126–134.

[51] C. Dwork, *Differential privacy: A survey of results*, in International conference on theory and applications of models of computation, Springer, 2008, pp. 1–19.

[52] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, *Our data, ourselves: Privacy via distributed noise generation*, in Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2006, pp. 486–503.

[53] C. Dwork, F. McSherry, K. Nissim, and A. Smith, *Calibrating noise to sensitivity in private data analysis*, in Theory of cryptography conference, Springer, 2006, pp. 265–284.

[54] C. Dwork, A. Roth, et al., *The algorithmic foundations of differential privacy.*, Found. Trends Theor. Comput. Sci., 9 (2014), pp. 211–407.

[55] K. El Makkaoui, A. Ezzati, A. Beni-Hssane, and S. Ouhmad, *Fast cloud–paillier homomorphic schemes for protecting confidentiality of sensitive*

data in cloud computing, Journal of Ambient Intelligence and Humanized Computing, (2019), pp. 1–10.

[56] H. Fang and Q. Qian, *Privacy preserving machine learning with homomorphic encryption and federated learning*, Future Internet, 13 (2021), p. 94.

[57] D. Feldman, A. Fiat, H. Kaplan, and K. Nissim, *Private coresets*, in Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 361–370.

[58] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, *Certifying and removing disparate impact*, in proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp. 259–268.

[59] M. Fredrikson, S. Jha, and T. Ristenpart, *Model inversion attacks that exploit confidence information and basic countermeasures*, in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, pp. 1322–1333.

[60] K. Gai, M. Qiu, and H. Zhao, *Privacy-preserving data encryption strategy for big data in mobile cloud computing*, IEEE Transactions on Big Data, (2017).

[61] C. Gentry, *Fully homomorphic encryption using ideal lattices*, in Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 169–178.

[62] C. Gentry, S. Halevi, and N. P. Smart, *Better bootstrapping in fully homomorphic encryption*, in International Workshop on Public Key Cryptography, Springer, 2012, pp. 1–16.

[63] ——, *Fully homomorphic encryption with polylog overhead*, in Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2012, pp. 465–482.

[64] I. Giacomelli, S. Jha, M. Joye, C. D. Page, and K. Yoon, *Privacy-preserving ridge regression with only linearly-homomorphic encryption*, in International Conference on Applied Cryptography and Network Security, Springer, 2018, pp. 243–261.

[65] C. Gicquel, *MIP models and exact methods for the Discrete Lot-sizing and Scheduling Problem with sequence-dependent changeover costs and times*, PhD thesis, Ecole Centrale Paris, 2008.

[66] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, *Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy*, in International conference on machine learning, PMLR, 2016, pp. 201–210.

[67] R. E. Goldschmidt, *Applications of division by convergence*, PhD thesis, Massachusetts Institute of Technology, 1964.

[68] M. Gong, Y. Xie, K. Pan, K. Feng, and A. K. Qin, *A survey on differentially private machine learning*, IEEE Computational Intelligence Magazine, 15 (2020), pp. 49–64.

[69] N. Z. Gong, A. Talwalkar, L. Mackey, L. Huang, E. C. R. Shin, E. Stefanov, E. Shi, and D. Song, *Joint link prediction and attribute inference using a social-attribute network*, ACM Transactions on Intelligent Systems and Technology (TIST), 5 (2014), pp. 1–20.

[70] R. M. Gower and P. Richtárik, *Linearly convergent randomized iterative methods for computing the pseudoinverse*, arXiv preprint arXiv:1612.06255, (2016).

[71] T. Graepel, K. Lauter, and M. Naehrig, *Ml confidential: Machine learning on encrypted data*, in International Conference on Information Security and Cryptology, Springer, 2012, pp. 1–21.

[72] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, *Covariate shift by kernel mean matching*, Dataset shift in machine learning, 3 (2009), p. 5.

[73] J. Guckenheimer and P. Holmes, *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, vol. 42, Springer Science & Business Media, 2013.

[74] L. Guo, X. Ying, and X. Wu, *On attribute disclosure in randomization based privacy preserving data publishing*, in 2010 IEEE International Conference on Data Mining Workshops, IEEE, 2010, pp. 466–473.

[75] S. Halevi, *Homomorphic encryption*, in Tutorials on the Foundations of Cryptography, Springer, 2017, pp. 219–276.

[76] S. Halevi and V. Shoup, *Bootstrapping for helib*, in Annual International conference on the theory and applications of cryptographic techniques, Springer, 2015, pp. 641–670.

[77] R. Hall, S. E. Fienberg, and Y. Nardi, *Secure multiple linear regression based on homomorphic encryption*, Journal of Official Statistics, 27 (2011), p. 669.

[78] K. Han and D. Ki, *Better bootstrapping for approximate homomorphic encryption*, in Topics in Cryptology – CT-RSA 2020, S. Jarecki, ed., Cham, 2020, Springer International Publishing, pp. 364–390.

[79] M. Hardt, E. Price, and N. Srebro, *Equality of opportunity in supervised learning*, in Advances in neural information processing systems, 2016, pp. 3315–3323.

[80] J. A. Hartigan, *Clustering algorithms*, John Wiley & Sons, Inc., 1975.

[81] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, *Privacy-preserving machine learning as a service.*, Proc. Priv. Enhancing Technol., 2018 (2018), pp. 123–142.

[82] M. W. Hirsch, *Differential topology*, vol. 33, Springer Science & Business Media, 2012.

[83] H. Huang, Y. Wang, and H. Zong, *Support vector machine classification over encrypted data*, Applied Intelligence, 52 (2022), pp. 5938–5948.

[84] J.-J. Huang, G.-H. Tzeng, and C.-S. Ong, *Marketing segmentation using support vector clustering*, Expert systems with applications, 32 (2007), pp. 313–317.

[85] Z. Huang and J. Liu, *Optimal differentially private algorithms for k-means clustering*, in Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, 2018, pp. 395–408.

[86] J. J. Hull, *A database for handwritten text recognition research*, IEEE Transactions on pattern analysis and machine intelligence, 16 (1994), pp. 550–554.

[87] H. Imtiaz and A. D. Sarwate, *Symmetric matrix perturbation for differentially-private principal component analysis*, in 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2016, pp. 2339–2343.

[88] M. Jagielski, M. Kearns, J. Mao, A. Oprea, A. Roth, S. Sharifi-Malvajerdi, and J. Ullman, *Differentially private fair learning*, in International Conference on Machine Learning, PMLR, 2019, pp. 3000–3008.

[89] J. Janai, F. Güney, A. Behl, A. Geiger, et al., *Computer vision for autonomous vehicles: Problems, datasets and state of the art*, Foundations and Trends® in Computer Graphics and Vision, 12 (2020), pp. 1–308.

[90] R. Jans and Z. Degraeve, *Modeling industrial lot sizing problems: a review*, International Journal of Production Research, 46 (2008), pp. 1619–1643.

[91] J. Jia and N. Z. Gong, *Attriguard: A practical defense against attribute inference attacks via adversarial machine learning*, in 27th {USENIX} Security Symposium ({USENIX} Security 18), 2018, pp. 513–529.

[92] H. T. Jongen, P. Jonker, and F. Twilt, *Nonlinear optimization in IRN*, P. Lang Publishing Co., 1987.

[93] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, {*GAZELLE*}*: A low latency framework for secure neural network inference*, in 27th USENIX Security Symposium (USENIX Security 18), 2018, pp. 1651–1669.

[94] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren, *Secure, privacy-preserving and federated machine learning in medical imaging*, Nature Machine Intelligence, (2020), pp. 1–7.

[95] G. Kamath, O. Sheffet, V. Singhal, and J. Ullman, *Differentially private algorithms for learning mixtures of separated gaussians*, Advances in Neural Information Processing Systems, 32 (2019), pp. 168–180.

[96] S. Kariyappa and M. K. Qureshi, *Defending against model stealing attacks with adaptive misinformation*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 770–778.

[97] M. Kearns, Y. Mansour, and A. Y. Ng, *An information-theoretic analysis of hard and soft assignment methods for clustering*, in Learning in graphical models, Springer, 1998, pp. 495–520.

[98] H. K. Khalil, *1992, nonlinear systems, macmillan, new york.*

[99] ——, *Nonlinear systems third edition*, Patience Hall, 115 (2002).

[100] N. Kilbertus, A. Gascón, M. Kusner, M. Veale, K. Gummadi, and A. Weller, *Blind justice: Fairness with encrypted sensitive attributes*, in International Conference on Machine Learning, PMLR, 2018, pp. 2630–2639.

[101] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, *Logistic regression model training based on the approximate homomorphic encryption*, BMC medical genomics, 11 (2018), pp. 23–31.

[102] K. Kim, Y. Son, and J. Lee, *Voronoi cell-based clustering using a kernel support*, IEEE Transactions on Knowledge and Data Engineering, 27 (2014), pp. 1146–1156.

[103] M. Kim, J. Lee, L. Ohno-Machado, and X. Jiang, *Secure and differentially private logistic regression for horizontally distributed data*, IEEE Transactions on Information Forensics and Security, 15 (2019), pp. 695–710.

[104] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, *Secure logistic regression based on homomorphic encryption: Design and evaluation*, JMIR medical informatics, 6 (2018), p. e19.

[105] M. Kosinski, D. Stillwell, and T. Graepel, *Private traits and attributes are predictable from digital records of human behavior*, Proceedings of the national academy of sciences, 110 (2013), pp. 5802–5805.

[106] D. Lee, K.-H. Jung, and J. Lee, *Constructing sparse kernel machines using attractors*, IEEE transactions on neural networks, 20 (2009), pp. 721–729.

[107] D. Lee and J. Lee, *Dynamic dissimilarity measure for support-based clustering*, IEEE Transactions on Knowledge and Data Engineering, 22 (2009), pp. 900–905.

[108] D. Lee, J. Lee, and Y.-G. Yoon, *A quadratic string adapted barrier exploring method for locating transition states*, Computer physics communications, 177 (2007), pp. 218–218.

[109] J. Lee and H.-D. Chiang, *A dynamical trajectory-based methodology for systematically computing multiple optimal solutions of general nonlinear programming problems*, IEEE Transactions on Automatic Control, 49 (2004), pp. 888–899.

[110] J. Lee and D. Lee, *Dynamic characterization of cluster structures for robust and inductive support vector clustering*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (2006), pp. 1869–1874.

[111] W. Lee, H. Kim, and J. Lee, *Compact class-conditional domain invariant learning for multi-class domain adaptation*, Pattern Recognition, 112 (2021), p. 107763.

[112] W. Lee, H. Ko, J. Byun, T. Yoon, and J. Lee, *Fair clustering with fair correspondence distribution*, Information Sciences, 581 (2021), pp. 155–178.

[113] B. Li and D. Micciancio, *On the security of homomorphic encryption on approximate numbers*, in Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2021, pp. 648–677.

[114] N. Li, T. Li, and S. Venkatasubramanian, *t-closeness: Privacy beyond k-anonymity and l-diversity*, in 2007 IEEE 23rd International Conference on Data Engineering, IEEE, 2007, pp. 106–115.

[115] Q. Liu, H. Shen, and Y. Sang, *A privacy-preserving data publishing method for multiple numerical sensitive attributes via clustering and multi-sensitive bucketization*, in 2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming, ieee, 2014, pp. 220–223.

[116] L. Lombardo and P. M. Mai, *Presenting logistic regression-based landslide susceptibility results*, Engineering geology, 244 (2018), pp. 14–24.

[117] Q. Lou, B. Feng, G. Charles Fox, and L. Jiang, *Glyph: Fast and accurately training deep neural networks on encrypted data*, Advances in Neural Information Processing Systems, 33 (2020), pp. 9193–9202.

[118] Q. Lou, B. Feng, G. C. Fox, and L. Jiang, *Glyph: Fast and accurately training deep neural networks on encrypted data*, arXiv preprint arXiv:1911.07101, (2019).

[119] Q. Lou and L. Jiang, *She: A fast and accurate deep neural network for encrypted data*, arXiv preprint arXiv:1906.00148, (2019).

[120] Q. Lou, W.-j. Lu, C. Hong, and L. Jiang, *Falcon: fast spectral inference on encrypted data*, Advances in Neural Information Processing Systems, 33 (2020), pp. 2364–2374.

[121] ——, *Falcon: Fast spectral inference on encrypted data*, in Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds., vol. 33, Curran Associates, Inc., 2020, pp. 2364–2374.

[122] L. Ma and B. Sun, *Machine learning and ai in marketing–connecting computing power to human insights*, International Journal of Research in Marketing, 37 (2020), pp. 481–504.

[123] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, *l-diversity: Privacy beyond k-anonymity*, ACM Transactions on Knowledge Discovery from Data (TKDD), 1 (2007), pp. 3–es.

[124] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, *Powerspy: Location tracking using mobile device power analysis*, in 24th {USENIX} Security Symposium ({USENIX} Security 15), 2015, pp. 785–800.

[125] F. Mireshghallah, M. Taram, P. Vepakomma, A. Singh, R. Raskar, and H. Esmaeilzadeh, *Privacy in deep learning: A survey*, arXiv preprint arXiv:2004.12254, (2020).

[126] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, *Delphi: A cryptographic inference service for neural networks*, in 29th USENIX Security Symposium (USENIX Security 20), 2020, pp. 2505–2522.

[127] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*, MIT press, 2018.

[128] T. Morshed, D. Alhadidi, and N. Mohammed, *Parallel linear regression on encrypted data*, in 2018 16th Annual Conference on Privacy, Security and Trust (PST), IEEE, 2018, pp. 1–5.

[129] L. Ni, C. Li, X. Wang, H. Jiang, and J. Yu, *Dp-mcdbscan: Differential privacy preserving multi-core dbscan clustering for network user data*, IEEE Access, 6 (2018), pp. 21053–21063.

[130] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, *Privacy-preserving ridge regression on hundreds of millions of records*, in 2013 IEEE Symposium on Security and Privacy, IEEE, 2013, pp. 334–348.

[131] K. Nissim, S. Raskhodnikova, and A. Smith, *Smooth sensitivity and sampling in private data analysis*, in Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, 2007, pp. 75–84.

[132] T. Orekondy, B. Schiele, and M. Fritz, *Knockoff nets: Stealing functionality of black-box models*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4954–4963.

[133] J. Otterbacher, *Inferring gender of movie reviewers: exploiting writing style, content and metadata*, in Proceedings of the 19th ACM international conference on Information and knowledge management, 2010, pp. 369–378.

[134] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes*, in International conference on the theory and applications of cryptographic techniques, Springer, 1999, pp. 223–238.

[135] R. Palais and S. Smale, *A generalized morse theory*, in The Collected Papers of Stephen Smale: Volume 2, 2000, pp. 503–510.

[136] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, *Practical black-box attacks against machine learning*, in Proceedings of the 2017 ACM on Asia conference on computer and communications security, 2017, pp. 506–519.

[137] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, *Scalable private learning with pate*, arXiv preprint arXiv:1802.08908, (2018).

[138] M. Park, J. Foulds, K. Choudhary, and M. Welling, *Dp-em: Differentially private expectation maximization*, in Artificial Intelligence and Statistics, PMLR, 2017, pp. 896–904.

[139] S. Park, J. Byun, and J. Lee, *Privacy-preserving fair learning of support vector machine with homomorphic encryption*, in Proceedings of the ACM Web Conference 2022, 2022, pp. 3572–3583.

[140] S. Park, J. Byun, J. Lee, J. H. Cheon, and J. Lee, *He-friendly algorithm for privacy-preserving svm training*, IEEE Access, 8 (2020), pp. 57414–57425.

[141] S. Park, J. Hah, and J. Lee, *Inductive ensemble clustering using kernel support matching*, Electronics Letters, 53 (2017), pp. 1625–1626.

[142] M. A. Pathak and B. Raj, *Large margin gaussian mixture models with differential privacy*, IEEE Transactions on dependable and secure computing, 9 (2012), pp. 463–469.

[143] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie, *Generative adversarial perturbations*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4422–4431.

[144] G. Qiu, X. Gui, and Y. Zhao, *Privacy-preserving linear regression on distributed data by homomorphic encryption and data masking*, IEEE Access, 8 (2020), pp. 107601–107613.

[145] Y. Rahulamathavan, R. C.-W. Phan, S. Veluru, K. Cumanan, and M. Rajarajan, *Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud*, IEEE Transactions on Dependable and Secure Computing, 11 (2013), pp. 467–479.

[146] C. E. Rasmussen, *Gaussian processes in machine learning*, in Summer school on machine learning, Springer, 2003, pp. 63–71.

[147] R. L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, 21 (1978), pp. 120–126.

[148] A. S. Sabau, *Survey of clustering based financial fraud detection research*, Informatica Economica, 16 (2012), p. 110.

[149] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, *Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models*, arXiv preprint arXiv:1806.01246, (2018).

[150] A. Samé, C. Ambroise, and G. Govaert, *An online classification em algorithm based on the mixture model*, Statistics and Computing, 17 (2007), pp. 209–218.

[151] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, *Estimating the support of a high-dimensional distribution*, Neural computation, 13 (2001), pp. 1443–1471.

[152] M. Seeger, *Learning with labeled and unlabeled data*, tech. rep., 2000.

[153] K. Shankar, M. Elhoseny, R. S. Kumar, S. Lakshmanaprabu, and X. Yuan, *Secret image sharing scheme with encrypted shadow images using optimal homomorphic encryption technique*, Journal of Ambient Intelligence and Humanized Computing, 11 (2020), pp. 1821–1833.

[154] M. Shechner, O. Sheffet, and U. Stemmer, *Private k-means clustering with stability assumptions*, in International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2518–2528.

[155] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke, *Personalized recommendation in social tagging systems using hierarchical clustering*, in Proceedings of the 2008 ACM conference on Recommender systems, 2008, pp. 259–266.

[156] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, *Membership inference attacks against machine learning models*, in 2017 IEEE symposium on security and privacy (SP), IEEE, 2017, pp. 3–18.

[157] S. Smale, *On gradient dynamical systems*, Annals of Mathematics, (1961), pp. 199–206.

[158] N. P. Smart and F. Vercauteren, *Fully homomorphic simd operations*, Designs, codes and cryptography, 71 (2014), pp. 57–81.

[159] G. Song, Y. Zhou, H. Liu, G. Wen, et al., *A privacy inference model based on attribute graph*, in 2018 International Conference on Networking and Network Applications (NaNA), IEEE, 2018, pp. 97–101.

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

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

[162] E. Suhubi, *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields: Applied mathematical science, vol. 42, j. guckenheimer and p. holmes, springer-verlag, new york, berlin, heidelberg, tokyo (1983). xvi+ 453 pp., 206 figs, dm 104*, 1988.

[163] M. Sumathi and S. Sangeetha, *Enhanced elliptic curve cryptographic technique for protecting sensitive attributes in cloud storage*, in 2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), IEEE, 2018, pp. 1–5.

[164] X. Sun, P. Zhang, J. K. Liu, J. Yu, and W. Xie, *Private machine learning classification based on fully homomorphic encryption*, IEEE Transactions on Emerging Topics in Computing, (2018).

[165] M. Svensén and C. M. Bishop, *Pattern recognition and machine learning*, 2007.

[166] L. Sweeney, *k-anonymity: A model for protecting privacy*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10 (2002), pp. 557–570.

[167] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, *Stealing machine learning models via prediction {APIs}*, in 25th USENIX security symposium (USENIX Security 16), 2016, pp. 601–618.

[168] A. Ultsch, *Clustering wih som: U* c*, in Proc. Workshop on Self-Organizing Maps (Jan. 2005)(zitiert auf S. 53), 2005.

[169] P. Voigt and A. Von dem Bussche, *The eu general data protection regulation (gdpr)*, A Practical Guide, 1st Ed., Cham: Springer International Publishing, (2017).

[170] Q. Wang and X. Shi, *(a, d)-diversity: Privacy protection based on l-diversity*, in 2009 WRI World Congress on Software Engineering, vol. 3, IEEE, 2009, pp. 367–372.

[171] Wikipedia contributors, *Facebook–cambridge analytica data scandal — Wikipedia, the free encyclopedia*, 2020. [Online; accessed 28-October-2020].

[172] L. A. Wolsey, *Integer programming*, Wiley, 1998.

[173] W.-m. Wu and H.-k. Huang, *A dp-dbscan clustering algorithm based on differential privacy preserving*, Computer Engineering and Science, 37 (2015), pp. 830–834.

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

[175] Xpress, *Xpress 8.0.* `http://www.fico.com/en`, 2016.

[176] K. Xu, H. Yue, L. Guo, Y. Guo, and Y. Fang, *Privacy-preserving machine learning algorithms for big data systems*, in 2015 IEEE 35th international conference on distributed computing systems, IEEE, 2015, pp. 318–327.

[177] R. Xu, N. Baracaldo, and J. Joshi, *Privacy-preserving machine learning: Methods, challenges and directions*, arXiv preprint arXiv:2108.04417, (2021).

[178] Q. Yang, Y. Liu, T. Chen, and Y. Tong, *Federated machine learning: Concept and applications*, ACM Transactions on Intelligent Systems and Technology (TIST), 10 (2019), pp. 1–19.

[179] A. C. Yao, *Protocols for secure computations*, in 23rd annual symposium on foundations of computer science (sfcs 1982), IEEE, 1982, pp. 160–164.

[180] A. C.-C. Yao, *How to generate and exchange secrets*, in 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), IEEE, 1986, pp. 162–167.

[181] T. Yoon, J. Lee, and W. Lee, *Joint transfer of model knowledge and fairness over domains using wasserstein distance*, IEEE Access, 8 (2020), pp. 123783–123798.

[182] Q. Yu, Y. Luo, C. Chen, and X. Ding, *Outlier-eliminated k-means clustering algorithm based on differential privacy preservation*, Applied Intelligence, 45 (2016), pp. 1179–1191.

[183] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, *Learning fair representations*, in International Conference on Machine Learning, 2013, pp. 325–333.

[184] Y. Zhang and J. Han, *Differential privacy fuzzy c-means clustering algorithm based on gaussian kernel function*, Plos one, 16 (2021), p. e0248737.

[185] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, *The secret revealer: Generative model-inversion attacks against deep neural networks*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 253–261.

[186] E. Zheleva and L. Getoor, *To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles*, in Proceedings of the 18th international conference on World wide web, 2009, pp. 531–540.

국문초록

최근 인공지능의 성공에는 여러 가지 요인이 있으나, 새로운 알고리즘의 개발과 정제된 데이터 양의 기하급수적인 증가로 인한 영향이 크다. 따라서 기계학습 모델과 데이터는 실재적 가치를 가지게 되며, 현실 세계에서 개인 또는 기업은 학습된 모델 또는 학습에 사용할 데이터를 제공함으로써 이익을 얻을 수 있다. 그러나, 데이터 또는 모델의 공유는 개인의 민감 정보를 유출함으로써 프라이버시의 침해로 이어질 수 있다는 사실이 밝혀지고 있다.

본 논문의 목표는 민감 정보를 보호할 수 있는 프라이버시 보존 기계학습 방법론을 개발하는 것이다. 이를 위해 최근 활발히 연구되고 있는 두 가지 프라이버시 보존 기술, 즉 동형 암호와 차분 프라이버시를 사용한다. 먼저, 동형 암호는 암호화된 데이터에 대해 기계학습 알고리즘을 적용 가능하게 함으로써 데이터의 프라이버시를 보호할 수 있다. 그러나 동형 암호를 활용한 연산은 기존의 연산에 비해 매우 큰 연산 시간을 요구하므로 효율적인 알고리즘을 구성하는 것이 중요하다. 효율적인 연산을 위해 우리는 두 가지 접근법을 사용한다. 첫 번째는 학습 단계에서의 연산량을 줄이는 것이다. 학습 단계에서부터 동형 암호를 적용하면 학습 데이터의 프라이버시를 함께 보호할 수 있으므로 추론 단계에서만 동형 암호를 적용하는 것에 비해 프라이버시의 범위가 넓어지지만, 그만큼 연산량이 늘어난다. 본 논문에서는 일부 가장 중요한 정보만을 암호화함으로써 학습 단계를 효율적으로 하는 방법론을 제안한다. 구체적으로, 일부 민감 변수가 암호화되어 있을 때 연산량을 매우 줄일 수 있는 릿지 회귀 알고리즘을 개발한다. 또한 개발된 알고리즘을 확장시켜 동형 암호 친화적이지 않은 파라미터 탐색 과정을 최대한 제거할 수 있는 새로운 로지스틱 회귀 알고리즘을 함께 제안한다. 효율적인 연산을 위한 두 번째 접근법은 동형 암호를 기계학습의 추론 단계에서만 사용하는 것이다. 이를 통해 시험

데이터의 직접적인 노출을 막을 수 있다. 본 논문에서는 서포트 벡터 군집화 모델에 대한 동형 암호 친화적 추론 방법을 제안한다.

동형 암호는 여러 가지 위협에 대해서 데이터와 모델 정보를 보호할 수 있으나, 학습된 모델을 통해 새로운 데이터에 대한 추론 서비스를 제공할 때 추론 결과로부터 모델과 학습 데이터를 보호하지 못한다. 연구를 통해 공격자가 자신이 가진 데이터와 그 데이터에 대한 추론 결과만을 이용하여 이용하여 모델과 학습 데이터에 대한 정보를 추출할 수 있음이 밝혀지고 있다. 예를 들어, 공격자는 특정 데이터가 학습 데이터에 포함되어 있는지 아닌지를 추론할 수 있다. 차분 프라이버시는 학습된 모델에 대한 특정 데이터 샘플의 영향을 줄임으로써 이러한 공격에 대한 방어를 보장하는 프라이버시 기술이다. 차분 프라이버시는 프라이버시의 수준을 정량적으로 표현함으로써 원하는 만큼의 프라이버시를 충족시킬 수 있지만, 프라이버시를 충족시키기 위해서는 알고리즘에 그만큼의 무작위성을 더해야 하므로 모델의 성능을 떨어뜨린다. 따라서, 본문에서는 모스 이론을 이용하여 차분 프라이버시 군집화 방법론의 프라이버시를 유지하면서도 그 성능을 끌어올리는 새로운 방법론을 제안한다.

본 논문에서 개발하는 프라이버시 보존 기계학습 방법론은 각기 다른 수준에서 프라이버시를 보호하며, 따라서 상호 보완적이다. 제안된 방법론들은 하나의 통합 시스템을 구축하여 기계학습이 개인의 민감 정보롤 보호해야 하는 여러 분야에서 더욱 널리 사용될 수 있도록 하는 기대 효과를 가진다.

주요어: 프라이버시 보존 기계학습, 동형 암호, 차분 프라이버시

학번: 2017-28535