Fostering Trustworthiness in Machine Learning Algorithms

A

Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Mengdi Huai

August 2022

APPROVAL SHEET

This

Dissertation

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Mengdi Huai

This Dissertation has been read and approved by the examing committee:

Advisor: Dr. Aidong Zhang

Advisor:

Committee Member: Dr. Hongning Wang

Committee Member: Dr. Jundong Li

Committee Member: Dr. Stefan Bekiranov

Committee Member: Dr. Haifeng Xu

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

August 2022

Acknowledgements

I would like to thank all the people who gave me tremendous support and help during my Ph.D. study. Without their support and help, this dissertation would not have been possible.

First and foremost, I would like to thank my advisor - Professor Aidong Zhang, for her invaluable supervision, continuous support, and persistent encouragement during my PhD study. I consider myself very fortunate to work with Professor Zhang, and her immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. She taught me not only the way of doing excellent research, but also how to write high quality research papers, mentor students, and find the good things in life. When I was in trouble, she was always there to provide help and support. Her passion in research, teaching, and student supervision has also motivated me to pursue a career in academia.

Next, I would like to express my deepest appreciation to my committee committee members, Professor Hongning Wang, Professor Jundong Li, Professor Haifeng Xu, and Professor Stefan Bekiranov, for their constructive comments and suggestions for my research projects. I would also like to extend my deepest gratitude to Professor Jinhui Xu, Professor Jing Gao, Professor Chunming Qiao, Professor Lu Su, and Professor Changyou Chen for valuable discussions, suggestions, and collaborations.

I would also like to express my great appreciation to all my colleagues and friends who supported me during these years. I have enjoyed the time we spent together, and they make my Ph.D. journey a pleasant and exciting one. In particular, I would like to thank Guangxu Xun, Liuyi Yao, Qiuling Suo, Hongfei Xue, Jinduo liu, Jianhui Sun, Kishlay Jha, Guangtao Zheng, Jiayi Chen, Jing Ma, Sanchit Sinha, Yaliang Li, Qi Li, Houping Xiao, Fenglong Ma, Chuanhao Li, Tianhang Zheng, Di Wang, Renqin Cai, Yi Zhu, Wenjun Jiang, and those I have not included their names here, for the collaborations and helpful advice.

I would like to thank my parents, Zhengui Huai and Xueqin Zhang, for loving and supporting me unconditionally. I would like to thank my sister and brother, Jiwei Huai and Jianwei Huai, for their constant help and encouragement. I would also like to thank my husband, Chenglin Miao, for all his sacrifices and being there all the time. Especially helpful to me is their belief in me, which has kept my spirits and motivation high during this process.

Abstract

Recent years have witnessed an explosion of works that develop and apply machine learning algorithms to build intelligent learning systems (e.g., medical decision systems and self-driving cars). However, traditional machine learning algorithms mainly focus on optimizing accuracy and efficiency, and they fail to consider how to foster trustworthiness in their design. Trustworthiness reflects the degree of a user's confidence that the deployed machine learning algorithms will operate as the user expects in the face of various circumstances such as human errors, system failures, and malicious attacks. The essential characteristics at the core of trustworthiness include model transparency, robustness against malicious attacks, and privacy preservation. Without fully studying the trustworthiness of the deployed machine learning algorithms, we will face a variety of devastating social and environmental consequences. In this dissertation, we take steps to study and address the untrustworthy issues in the design of machine learning algorithms. Specifically, we first propose several model interpretation methods that can give insights on machine learning models' working mechanisms by interpreting what they have learned and hence help increase the trust in model decisions. Then, we design both offensive and defensive strategies to investigate the security vulnerabilities of machine learning algorithms to malicious attacks. In addition, we design several effective privacy-preserving mechanisms for privately sharing data and machine learning models without leaking the sensitive information. Extensive experiments are conducted and presented to demonstrate the effectiveness of the proposed methods.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

We have entered into the data age where everything around us is connected to a data source, and everything in our lives is digitally recorded. For instance, the current electronic world has a wealth of various kinds of data, including the Electronic Health Records (EHRs), social media data, smartphone data, smart city data, cybersecurity data, spatiotemporal data, genomics data, and the Internet of Things (IoT) data. Extracting knowledge and useful insights from these wealthy data can be used for building smart decision-making systems in various relevant domains. For instance, to build personalized healthcare applications to improve health outcomes, the relevant healthcare data can be utilized; to build the efficient and intelligent cybersecurity systems, the relevant cybersecurity data can be exploited, and so on.

Particularly, machine learning have grown rapidly in recent years in the context of data analysis and computing that typically allows the applications to function in an intelligent manner. Machine learning is the essence of machine intelligence, and refers to the development of the systems that can automatically improve through experience and through the applications of real-time data. Specifically, machine learning consists of the techniques that can enable computers to discover different patterns in data by using flexible methods for modeling and variable selection and deliver artificial intelligence applications.

The field of machine learning has exploded in recent years and researchers have developed an enormous number of algorithms to choose from. Based on the nature of the "signal" (or "feedback") available to the learning systems, machine learning algorithms can be generally divided into the following three categories: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. In supervised learning, the training data

includes observations and labels, which represent some sort of true outcome or common human practice in reacting to the observations. In unsupervised learning, the training data only includes observations and no labels are given to the algorithm. In semi-supervised learning, a combination of a small amount of labeled data and a large amount of unlabeled data is used to train machine learning models. Thus, it falls between learning without supervision and learning with supervision. In reinforcement learning, the inputs are interactions with the real world and rewards accrued through those actions rather than a fixed dataset, and no predefined data are given and the agent learns by the trial-and-error method. In practice, machine learning models can be utilized to perform different tasks, e.g., classification analysis, regression analysis, data clustering, association rule learning, and feature engineering for dimensionality reduction.

Despite this great variety of machine learning models to choose from, they can all be distilled into three components. The three components that make a machine learning model are *representation*, *evaluation*, and *optimization*. Specifically, representation refers to the encoded representations of objects, such as a face being represented by features such as "eyes". Evaluation is used to distinguish good models from bad models. This evaluation is done using an evaluation function (e.g., accuracy and squared error). Optimization means updating the model parameters to minimize the value of loss function. Specifically, an optimization algorithm is used to fund the values of the parameters that minimize the error of the function when used to map inputs to outputs. This means that each time we fit a machine learning algorithm on a training dataset, we are solving an optimization problem.

Nowadays, machine learning becomes popular in various application areas, due to its learning capabilities from the experience and making intelligent decisions. The popular application areas of machine learning technology include image recognition, sentiment analysis, product recommendations, automating employee access control, marine wildlife preservation, language translation, language translation, and sustainable agriculture. In addition to these application areas, machine learning based models can also apply to several other domains such as bioinformatics, cheminformatics, computer networks, DNA sequence classifcation, economics and banking, robotics, advanced engineering, and so on.

1.1 Motivation

Although machine learning has been applied successfully in a wide variety of areas (e.g., disease diagnosis, autonomous driving, fraud detection, face recognition, and product recommendations), traditional machine learning algorithms mainly focus on optimizing accuracy, and they

fail to consider trustworthiness in their design.

- First, users usually treat a machine learning model as a black box because of its incomprehensible functions and complex working mechanism. However, the "black box" nature of the machine learning models may impede decision makers from trusting the predicted machine learning results, especially when the model is used for making critical decisions (e.g., medical disease diagnosis and autonomous deriving), because the consequences may be catastrophic if the predictions are acted upon blind faith. In general, transparent machine learning models have the advantage of faithfully reflecting the model behavior during the decision-making process, which helps users to examine whether a machine learning model has employed the true evidences instead of biases (that widely exist among training data) and reduces the likelihood of an error.

- Second, traditional approaches to machine learning usually assume that the training and testing data distributions are stationary and do not consider the trustworthiness in the presence of active adversaries. Recent studies have demonstrated that the trustworthiness in terms of security can be violated when motivated adversaries perturb test examples at testing time through evasion attacks, or inject well-crafted malicious instances into training data to induce learning errors through poisoning attacks.

- In addition, traditional machine learning methods also face serious privacy leakage risks. And the privacy of the sensitive private information refers to the trust that the sensitive information is not compromised or altered maliciously. In practice, privacy can be compromised when the shared data and machine learning models are exposed to an adversary. For example, in model inversion attacks, an adversary uses the shared model to make predictions of sensitive attributes (used as input to the model) of a target individual when some background information about the target individual is available.

Without fully studying all of the above trustworthiness aspects (i.e., transparency, security and privacy), we will face a variety of devastating social and environmental consequences, which are harmful to the connections between humans and the deployed real-world intelligent learning systems. In fact, the machine learning community also recognizes that all-hands efforts at various levels are needed to support and ensure the development of robust and trustworthy machine learning systems. Thus, there is a great need to ensure that the real-world intelligent learning systems that rely on machine learning are trustworthy in terms of transparency, security and privacy preservation.

1.2 Dissertation Overview

In this dissertation, towards the objective of fostering trustworthiness in machine learning algorithms, we take steps to address the untrustworthy issues when machine learning algorithms are applied in different real-world applications. According to the studied problems,we organize the dissertation with three parts: 1) research on enabling transparency for predictive machine learning models to address the problem of interpreting the predictions of black-box machine learning models, 2) research on investigating the security vulnerability and robustness of machine learning algorithms in the malicious environments, and 3) research on designing effective privacy-preserving mechanisms for privately sharing the information. In this section, we provide an overview of each work.

Enabling Transparency in Predictive Models

As we know, machine learning models have demonstrated great success in learning complex patterns that enable them to make predictions about unobserved data. However, in addition to using machine learning models for prediction, academic researchers and industrial practitioners are facing challenges that demand more transparent and explainable systems for better understanding the inner working mechanisms of machine learning models. The concerns about the black-box nature of machine learning models impede decision-makers from trusting the model predictions in real-world applications. For example, an advanced self-driving car equipped with various machine learning algorithms does not brake or decelerate when confronting a stopped firetruck. This unexpected behavior may frustrate and confuse users, making them wonder why. Even worse, the wrong decisions could cause severe consequences if the car is driving at highway speeds and might ultimately crash into the firetruck. In general, transparent algorithms have the advantage of, potentially, more eyes on them, reducing the likelihood that an error or oversight results in a bad outcome.

Thus, we here dive into model interpretation to interpret what a model has learned. Specifically, we first utilize the feature importance scoring as a specific approach to address the problem of interpreting the predictions of black-box pairwise learning models, which involve pairs of instances as the input of the loss functions. We also provide theoretical analysis to show that the proposed adaptive interpretation method for pairwise learning is the unique interpretation solution for pairwise learning with the desired properties. In addition, to investigate how accurately an algorithm is able to predict outcome values for previously unseen data, we also prove a generalization error bound for deep metric learning, which is a group of techniques that aims to measure the similarity between the data samples. The derived generalization error bound for

deep metric learning can help explain the behaviors of existing deep metric learning models and guide the design of good neural networks for deep metric learning.

Studying Security Vulnerability and Robustness to Malicious Attacks

Malicious attacks exist because there are inherent security vulnerabilities in the underlying machine algorithms that attackers can exploit to breach security and make the system fail. Security threats can be anything that attackers take advantage of a vulnerability to breach security and negatively alter objects of interest. For security, an attacker's goal is to manipulate a machine learning system such that the system makes predictions as the attacker desires. An attacker can manipulate the training phase and/or the testing phase to achieve this goal.

In order to analyze attackers' actions in adversarial environments and thereby derive robust learning strategies, we here study the security vulnerability and robustness of machine learning methods. In particular, we first discover the vulnerability of the model interpretation methods for deep reinforcement learning to two different representative types of attacks, i.e., adversarial attacks and model poisoning attacks. Specifically, we first design an universal adversarial attack against deep reinforcement learning interpretations and then design a model poisoning attack, based on which the attacker can significantly alter the interpretation results while ensuring the stealthiness of the performed attacks. We also design a general defense mechanism to provide the guarantees of robustness of the interpretation results. In addition, we also design a novel robust and automatic self-explaining method, which can provide certified robustness guarantees for the generated concept-based automatic explanations.

Privacy-preserving Sharing of Sensitive Information

With the development of new emerging technologies, information sharing is becoming increasingly popular, and the shared information can fuel a variety of services from personalized healthcare recommendations to personalized product. As the scale of information sharing expands, however, there is also growing concerns about privacy and protecting sensitive information since the shared information usually contains sensitive information (e.g., regarding our health or our daily activities). If any adversary attack can be applied to learn any private and sensitive information, there is a privacy leakage. And the shared information is vulnerable to privacy attacks (e.g., membership inference attacks and model inversion attacks). With the increased concerns about privacy breaches, both data and model providers in machine learning systems have a growing desire for great confidentiality/privacy: data providers desire privacy of their shared data, while model providers desire confidentiality of their proprietary learning models as they represent intellectual property. The growing privacy concerns have been influencing data

owners and preventing them from achieving the maximum benefit of data sharing.

Motivated by this, we here propose novel privacy-preserving mechanisms to allow information providers to privately share the private sensitive information. In particular, we first use differential privacy techniques to keep the sensitive information of pairwise learning models private in both of the online and offline settings, while guaranteeing good generalization performance. Specifically, for each setting, we first propose a differentially private algorithm for the strongly convex loss functions, and then extend this algorithm to general convex loss functions by proposing another differentially private algorithm. Additionally, we also propose a novel privacy-aware synthesizing method for crowdsourced data, based on which the data collector can release users' data with strong privacy protection for their private information, while at the same time, the data analyzer can achieve good utility from the released data.

1.3 Dissertation Organization

In the next six chapters, I will elaborate on the aforementioned trustworthy mechanisms, shedding light on their design philosophy and desirable properties. Specifically,

- In Chapter 2, we investigate how to enable interpretation in pairwise learning. Specifically, we first propose a novel adaptive interpretation method for pairwise learning, based on which a vector of importance scores associated with the underlying features of a testing instance pair can be adaptively calculated, and these importance scores can be used to indicate which features make key contributions to the final prediction. Considering that this proposed pairwise interpretation method is computationally challenging, we further propose a novel robust approximation interpretation method for pairwise learning models. This proposed pairwise approximation interpretation method is not only much more efficient but also robust to data noise. We also conduct theoretical analysis to verify the effectiveness of the proposed interpretation methods for pairwise learning.

- In Chapter 3, to understand how good a learned deep metric learning model is able to perform on unseen data, we derive the generalization error bound for deep metric learning, which can give a comprehensive theoretical generalization analysis for the trained deep metric learning models and provide much important information about the practical performance of deep metric learning. In addition, based on the derived generalization bound, we propose a novel method to adaptively learn the retention rates for the deep metric learning models with dropout in a theoretically justified way. Compared with existing

deep metric learning works that require predefined retention rates, this proposed novel method can learn the retention rates in an optimal way and achieve better performance. We also conduct experiments on real-world datasets to verify the findings derived from the generalization error bound and demonstrate the effectiveness of the proposed adaptive deep metric learning method.

- In Chapter 4, we study the vulnerability of deep reinforcement learning interpretations to the malicious attacks. More specifically, in this chapter, we firstly present an universal adversarial attack against deep reinforcement learning interpretations, from which the attacker can add the crafted universal perturbation uniformly to the environment states in a maximum number of steps to incur minimal damage to the agent's end goal. Then, we design a model poisoning attack against deep reinforcement learning interpretations, which can significantly alter the interpretation results while incurring minor damage to the model performance. To enhance the robustness of deep reinforcement learning interpretations against malicious attacks, we also propose a general defense mechanism to increase the attack resistance of deep reinforcement learning interpretations against the malicious attacks. Both theoretical analysis and extensive experimental results are provided to demonstrate the effectiveness of our proposed approaches.

- In Chapter 5, we design a novel robust and automatic self-explaining method that can not only automatically provide the concept-based explanations without human interventions but also provide certified robustness guarantees for the generated concept-based explanations. Specifically, to free human from the tedious manual defining procedure, we first proposed a novel interpretability regularizer that guides the model to automatically extract the prototype-based concepts from the training data, which provide insights into representative patterns that are utilized by the model for classification. In addition, to promote certified robust interpretability, we also proposed a novel interval bound propagation based regularizer, which minimizes an upper bound on the maximum difference between any pair of explanation results when the input can be adversarially perturbed to provide verifiable robustness guarantees for the generated explanations. We also conducted experiments to demonstrate the effectiveness of the proposed method on real-world datasets.

- In Chapter 6, we consider the pairwise learning problems in both online and offline settings. For the online setting, we first propose an differential privacy algorithm for the

strongly convex loss functions, and then extend this proposed differential private algorithm to general convex loss functions by proposing another differentially private algorithm. For the offline setting, we also propose two differentially private algorithms for strongly convex loss functions and general convex loss functions, respectively, and then give their regret upper bounds. The experimental results on real-world datasets not only confirm our theoretical analysis but also demonstrate the effectiveness of the proposed algorithms in real-world applications.

- In Chapter 7, we propose a novel privacy-aware synthesizing method for crowdsourced data. Based on this method, the data collector can release the crowdsourced data with strong privacy protection for users' private information, while at the same time, the data analyzer can achieve good utility from the released data. We also conduct both theoretical analysis and extensive experiments on real-world datasets to verify the effectiveness of the proposed private synthesizing method.

Finally, in Chapter 8, we conclude the research contributions of this dissertation. Then, we discuss the directions for future research.

Part I

Enabling Transparency for Predictive Models

Chapter 2

Towards Interpretation of Pairwise Learning

2.1 Introduction

In recent years, there has been increasing interest in an important family of learning problems that is categorized as pairwise learning [1]. Different from the traditional pointwise learning (e.g., regression and classification) [2] where the loss function takes only individual instances as its input, pairwise learning involves pairs of instances as the input of its loss function. Comparing to pointwise learning, pairwise learning is more capable of modeling the relative relationship between pairs of instances, which has been demonstrated in many real-world applications. For example, in patient similarity learning, the learner (e.g., a doctor/hospital) can learn a clinically meaningful similarity metric to measure the proximity between a pair of patients through formulating the learning task as a pairwise learning problem [3]. Additionally, many other learning tasks can also be classified as pairwise learning, such as AUC maximization [4, 5], metric learning [6, 7, 8], bipartite ranking [9].

Despite its tremendous success in many real-world applications, pairwise learning still faces one challenging problem, i.e., the lack of transparency behind its behaviors, which makes it difficult for users to understand how particular decisions are made by the learned pairwise model. For instance, in the patient similarity learning task, the similarity metric is usually learned from a large amount of high dimensional and complex patient data. The learner can obtain the proximity between a pair of patients based on the learned metric, but he/she has no idea why the

metric reports such proximity. The "black box" nature of the learned pairwise models may impede users from trusting the predicted results, especially when the model is used for making critical decisions (e.g., medical diagnosis), because the consequences may be catastrophic if the predictions are acted upon blind faith. The lack of transparency behind pairwise learning models has hampered their further applications in real world. Thus, it is essential to investigate how to enable interpretaion in pairwise learning.

In this chapter, we aim to study feature importance scoring as a specific approach to the problem of interpreting the predictions of black-box pairwise models. Specifically, given a learned pairwise model and a testing instance pair, we hope to design an interpretation method that can generate a vector of importance scores associated with the underlying features of the testing instance pair, and enable these importance scores to indicate which features make key contributions to the final predicted result. There is now many interpretation methods that can score the importance of the input features for traditional pointwise learning models (e.g., classification models). Among them, the Shapley-value-based methods [10, 11, 12, 13, 14, 15, 16, 17] have drawn significant attention as they are the only methods that can provide theoretical guarantee. However, these methods cannot be directly used for pairwise models. First of all, to score the importance of a subset of input features, these methods usually need to pre-define a reference vector to mask the rest features. An implicit assumption in these methods is that all the testing instances use the same reference vector, which is unreasonable for pairwise models. When interpreting the predictions made by pairwise models, if both instances in the testing pair use the same reference vector, the relationship between them will be largely affected (e.g., may make them more similar) and wrong prediction may be generated. Additionally, existing interpretation methods for pointwise learning usually assume that the input features are nearly independent. However, in practice, the features may be correlated with each other and the correlation can also affect the predictions made by the models [18].

To address the above challenges, in this chapter, we first propose a novel adaptive Shapley-value-based interpretation method for pairwise models (**ASIPair**), which not only takes into account feature correlations but also can adaptively calculate the importance scores of the underlying features for each testing instance pair. We also provide theoretical analysis to show that the proposed adaptive method is the unique solution with the desired properties. Considering that Shapley-value-based methods are usually computationally challenging, we further propose a robust approximation interpretation method for pairwise models (**RAIPair**), which is motivated by the fact that not all features are important and only a subset of features contain the discriminative information for the final predicted result. The proposed approximated

interpretation method does not make any assumptions on the underlying feature structure and is also robust to data noise. To the best of our knowledge, we are the first to investigate how to enable interpretation in pairwise learning. Both theoretical analysis and extensive experiments demonstrate the effectiveness of the proposed interpretation methods for pairwise learning.

2.2 Methodology

In this section, we describe the proposed interpretation methods for pairwise models. Specifically, we first propose an adaptive Shapley-value-based interpretation method (called ASIPair) with the consideration of feature correlations. Considering that Shapley-value-based methods are usually computationally challenging, we then propose a robust approximation interpretation method (called RAIPair).

2.2.1 Adaptive Interpretation Method for Pairwise Models

The importance of each feature in $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ can be reflected by its contribution to the final predicted result. For any given subset $T \subset [D] = \{1, 2, ..., D\}$, we use $\boldsymbol{x}_i^T = \{x_{i,t}, t \in T\}$ to denote the associated sub-vector of features, where $x_{i,t}$ denotes the t-th element in $\boldsymbol{x}_i$. Let $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^T, \boldsymbol{x}_j^T]$ be the induced expected conditional prediction for the testing instance pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ when it is restricted to using only the sub-vectors $\boldsymbol{x}_i^T$ and $\boldsymbol{x}_j^T$. Then, for a given subset $T \subset [D] \setminus \{d\}$, the marginal contribution of the d-th feature to T (joining the subset T) can be calculated as follows

$$\begin{aligned}\Delta_d(\boldsymbol{x}_i, \boldsymbol{x}_j, T, \zeta) =& \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T\cup\{d\}}, \boldsymbol{x}_j^{T\cup\{d\}}] \\ &- \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^T, \boldsymbol{x}_j^T].\end{aligned} \tag{2.1}$$

To obtain $\Delta_d(\boldsymbol{x}_i, \boldsymbol{x}_j, T, \zeta)$, we need to calculate the two expected pairwise conditional functions $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T\cup\{d\}}, \boldsymbol{x}_j^{T\cup\{d\}}]$ and $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^T, \boldsymbol{x}_j^T]$.

As described in the introduction section, existing interpretation methods developed for traditional pointwise learning models cannot be directly used here to calculate $\Delta_d(\boldsymbol{x}_i, \boldsymbol{x}_j, T, \zeta)$ for pairwise learning. There is an implicit assumption in these methods that all the testing instances use the same pre-defined reference vector that is used for replacing $x_{i,t}$ ($t \in [D] \setminus T$) when measuring the contribution of $\boldsymbol{x}_i^T$ to the prediction, which is unreasonable for pairwise models. For pairwise models, if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ use the same reference vector, we cannot obtain reasonable $\Delta_d(\boldsymbol{x}_i, \boldsymbol{x}_j, T, \zeta)$, because the relationship between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ will be largely affected

and wrong prediction may be generated. Furthermore, these methods assume that the input features are nearly independent. However, in practice, the features are usually correlated with each other and the correlations can also affect the predicted result. To address the above challenges, we propose the following calculation method for $\Delta_d(\boldsymbol{x}_i, \boldsymbol{x}_j, T, \zeta)$.

Since the calculation procedures for $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T\cup\{d\}}, \boldsymbol{x}_j^{T\cup\{d\}}]$ and $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T}, \boldsymbol{x}_j^{T}]$ are similar, here we take $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T}, \boldsymbol{x}_j^{T}]$ as an example to describe the calculation procedure. We first rewrite the expected conditional pairwise function $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T}, \boldsymbol{x}_j^{T}]$ as follows

$$\begin{aligned}&\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T}, \boldsymbol{x}_j^{T}] = \\ &\int\int \zeta([\boldsymbol{x}_i^{\bar{T}}, \boldsymbol{x}_i^{T}], [\boldsymbol{x}_j^{\bar{T}}, \boldsymbol{x}_j^{T}]) P(\boldsymbol{x}_i^{\bar{T}}|\boldsymbol{x}_i^{T}) P(\boldsymbol{x}_j^{\bar{T}}|\boldsymbol{x}_j^{T}) d\boldsymbol{x}_i^{\bar{T}} d\boldsymbol{x}_j^{\bar{T}},\end{aligned} \tag{2.2}$$

where $\bar{T} = [D]\backslash T$ and $P(\boldsymbol{x}_i^{\bar{T}}|\boldsymbol{x}_i^{T})$ denotes the conditional distribution of $\boldsymbol{x}_i^{\bar{T}}$ given $\boldsymbol{x}_i^{T}$. $[\boldsymbol{x}_i^{\bar{T}}, \boldsymbol{x}_i^{T}]$ denotes the concatenation of $\boldsymbol{x}_i^{\bar{T}}$ and $\boldsymbol{x}_i^{T}$, i.e., $\boldsymbol{x}_i = [\boldsymbol{x}_i^{\bar{T}}, \boldsymbol{x}_i^{T}]$. To take the feature correlation into account, we propose to incorporate the covariance matrix that contains features' correlation information into the calculation process of the expected conditional pairwise function. Suppose the training set is denoted as $\{\boldsymbol{x}_k\}_{k=1}^{K}$, where K is the size of the training set. $\boldsymbol{x}_i^{T}$ and $\boldsymbol{x}_j^{T}$ can be transformed as

$$\tilde{\boldsymbol{x}}_i^{T} = \boldsymbol{\Omega}_T^{-1/2}(\boldsymbol{x}_i^{T} - \boldsymbol{\mu}_T), \quad \tilde{\boldsymbol{x}}_j^{T} = \boldsymbol{\Omega}_T^{-1/2}(\boldsymbol{x}_j^{T} - \boldsymbol{\mu}_T), \tag{2.3}$$

where $\boldsymbol{\mu}_T$ and $\boldsymbol{\Omega}_T$ denote the mean vector and covariance matrix of the set of sub-vectors $\{\boldsymbol{x}_k^{T}\}_{k=1}^{K}$ for training instances, respectively. The t-th element in $\boldsymbol{\mu}_T$ represents the mean value of the t-th feature over $\{\boldsymbol{x}_k^{T}\}_{k=1}^{K}$. Considering the fact that the training instance $\boldsymbol{x}_k = [\boldsymbol{x}_k^{\bar{T}}, \boldsymbol{x}_k^{T}]$ with $\boldsymbol{x}_k^{T}$ close to $\boldsymbol{x}_i^{T}$ is more informative when calculating $P(\boldsymbol{x}_i^{\bar{T}}|\boldsymbol{x}_i^{T})$, we then propose to use the training instances $\{\boldsymbol{x}_k\}_{k=1}^{K}$ to empirically calculate the pairwise conditional expectation $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T}, \boldsymbol{x}_j^{T}]$. Specifically, we first calculate the distance between $\boldsymbol{x}_i$ and each instance $\boldsymbol{x}_k$ in the training set $\{\boldsymbol{x}_k\}_{k=1}^{K}$ as

$$\begin{aligned}&Q_i^{T}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \\ &(\tilde{\boldsymbol{x}}_i^{T} - \boldsymbol{\Omega}_T^{-1/2}(\boldsymbol{x}_k^{T} - \boldsymbol{\mu}_T))'(\tilde{\boldsymbol{x}}_i^{T} - \boldsymbol{\Omega}_T^{-1/2}(\boldsymbol{x}_k^{T} - \boldsymbol{\mu}_T))/|T|.\end{aligned} \tag{2.4}$$

The distance between the testing instance $\boldsymbol{x}_j$ and the training instance $\boldsymbol{x}_k$ can be calculated in a similar way. Then, for each pair $(\boldsymbol{x}_i, \boldsymbol{x}_k)$ where $k \in [K]$, we calculate a weight $w_T(\boldsymbol{x}_i, \boldsymbol{x}_k) = \exp(-Q_i^{T}(\boldsymbol{x}_i, \boldsymbol{x}_k)/2\sigma^2)$, where σ is a smoothing parameter (the value is set as 0.2 in our experiment). After deriving all the weights $\{w_T(\boldsymbol{x}_i, \boldsymbol{x}_k)\}_{k=1}^{K}$, we sort these weights in an increasing order, and we use $\boldsymbol{x}_{k'}$ to denote the training instance corresponding to the k'-th element in the

ordered weight set. Similarly, we can derive the weights $\{w_T(\boldsymbol{x}_j, \boldsymbol{x}_k)\}_{k=1}^{K}$ for $\boldsymbol{x}_j$ and order them in an increasing order. Let $\boldsymbol{x}_{k''}$ be the training instance corresponding to the k''-th element in the ordered weight set for $\boldsymbol{x}_j$. Then, we can estimate $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^T, \boldsymbol{x}_j^T]$ as

$$\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^T, \boldsymbol{x}_j^T)] = \frac{1}{\sum_{k'=1}^{K_1} w_T(\boldsymbol{x}_i, \boldsymbol{x}_{k'}) \sum_{k''=1}^{K_1} w_T(\boldsymbol{x}_j, \boldsymbol{x}_{k''})} \{ \sum_{k'=1}^{K_1} w_T(\boldsymbol{x}_i, \boldsymbol{x}_{k'}) \cdot [\sum_{k''=1}^{K_1} w_T(\boldsymbol{x}_j, \boldsymbol{x}_{k''}) \zeta([\bar{\boldsymbol{x}_{k'}^T}, \boldsymbol{x}_i^T], [\bar{\boldsymbol{x}_{k''}^T}, \boldsymbol{x}_j^T])] \}, \tag{2.5}$$

where K_1 denotes the number of the selected training instances, and it can be decided as

$$K_1 = \arg\min_{L \in [K]} \{ \frac{\sum_{k'=1}^{L} w_T(\boldsymbol{x}_i, \boldsymbol{x}_{k'}) \sum_{k''=1}^{L} w_T(\boldsymbol{x}_j, \boldsymbol{x}_{k''})}{\sum_{k'=1}^{K} w_T(\boldsymbol{x}_i, \boldsymbol{x}_{k'}) \sum_{k''=1}^{K} w_T(\boldsymbol{x}_j, \boldsymbol{x}_{k''})} \geq \eta \}.$$

Here η is a pre-defined constant.

After calculating $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T \cup \{d\}}, \boldsymbol{x}_j^{T \cup \{d\}}]$ and $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^T, \boldsymbol{x}_j^T]$, we can then derive $\Delta_d(x_i, x_j, T, \zeta)$. Finally, taking all possible subset $T \subset [D] \setminus \{d\}$ into account, the contribution (i.e., importance score) of the d-th feature to the prediction of ζ on $(\boldsymbol{x_i}, \boldsymbol{x_j})$ is given as

$$\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) = \sum_{T \subset [D] \setminus \{d\}} \frac{|T|!(D - |T| - 1)!}{D!} \Delta_d(\boldsymbol{x}_i, \boldsymbol{x}_j, T, \zeta). \tag{2.6}$$

The above equation captures the average marginal contribution of the d-th feature by averaging $\Delta_d(\boldsymbol{x}_i, \boldsymbol{x}_j, T, \zeta)$ over all the possible subset T. The value of $\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)$ reflects the importance of the d-th feature to the final pairwise prediction. Based on this fact, we can identify which features in $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ make key contributions to the final predicted result.

An alternative way to calculate the contribution. Besides Eqn. (2.6), we also have another way to calculate the contribution of the d-th feature to the prediction of ζ on $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Let $\pi(D)$ be the set of all possible ordered permutations of the feature indices $\{1, 2, ..., D\}$. Let $\mathcal{O}$ be any permutation of the feature index $\{1, 2, ..., D\}$. For the permutation $\mathcal{O} \in \pi(D)$, we denote the set of features that precede d in $\mathcal{O}$ as $P_{\mathcal{O}}^d$. From Eqn. (2.6), we know that $\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)$ is the average marginal contribution of d to any coalition of D assuming that all orderings are equal. Another way to calculate $\phi_i(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)$ is averaging the marginal contributions of the d-th feature to the set of its predecessors, where the average value is taken over all permutations equally. Thus, we have

$$\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) = \frac{1}{D!} \sum_{\mathcal{O} \in \pi(D)} (\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d \cup \{d\}}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d \cup \{d\}}] - \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d}]). \tag{2.7}$$

Next, we provide the theoretical analysis to show that the proposed ASIPair is the unique pairwise interpretation method with the desired theoretical properties, which strongly motivates the use of ASIPair for reliable pairwise interpretations.

Theorem 1. *The proposed ASIPair is the unique solution that satisfies the following properties:*

(1) Efficiency. The sum of the marginal contributions of all input features is equal to the pairwise function value, i.e. $\sum_{d=1}^{D} \phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) = \zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

(2) Fairness. For all $T \subset \{1, 2, \cdots, D\} \setminus \{d_1, d_2\}$*, if* $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T\cup\{d_1\}}, \boldsymbol{x}_j^{T\cup\{d_1\}}] = \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T\cup\{d_2\}}, \boldsymbol{x}_j^{T\cup\{d_2\}}]$*, then we have* $\phi_{d_1}(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) = \phi_{d_2}(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)$.

(3) Dummy. For all subset $T \subset \{1, 2, \cdots, D\} \setminus \{d\}$*, if* $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T\cup\{d\}}, \boldsymbol{x}_j^{T\cup\{d\}}] = \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{T}, \boldsymbol{x}_j^{T}]$*, then we have* $\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) = 0$.

(4) Additivity. For any two pairwise decision functions ζ_1 *and* ζ_2*, we have that for each* $d \in [D]$*,* $\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta_1 + \zeta_2) = \phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta_1) + \phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta_2)$*, where* $(\zeta_1 + \zeta_2)$ *is defined by* $(\zeta_1 + \zeta_2)(\boldsymbol{x}_i, \boldsymbol{x}_j) = \zeta_1(\boldsymbol{x}_i, \boldsymbol{x}_j) + \zeta_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Note that all the four properties are reasonable in the context of pairwise learning. The efficiency property states that the total pairwise value $\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is divided among all of the features. This property makes it easier to compare features' contributions. The fairness property means that if two features always add the same marginal value to any subset to which they are added, they will be assigned equal contributions on the total pairwise value $\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The dummy property states that if a feature never adds any marginal value, the contribution value of this feature will be assigned with zero. The additivity property shows that the solution to the sum of two pairwise models (ζ_1 and ζ_2) must be the sum of what it assigns to each of the two pairwise models. Although the proposed ASIPair method can effectively score the importance of the input features on the predicted result and provide theoretical guarantee, the Shapley value approach makes it computationally challenging. The computation complexity of ASIPair (in terms of the pairwise model evaluations) on all features' contributions (i.e., $\{\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)\}_{d=1}^{D}$) is $O(D * 2^{D \log_2 D})$. To address this problem, we further propose a more efficient interpretation method for pairwise models in the next section.

2.2.2 Robust Approximation Interpretation Method for Pairwise Models

In this section, we propose a robust approximation interpretation method for pairwise models (RAIPair), which is motivated by that not all features are important and only a subset of features contain the discriminative information for the final predicted result. RAIPair is not only much more efficient than ASIPair but also robust to data noise.

Let $\boldsymbol{\phi} = (\phi_1(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta), \cdots, \phi_D(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)) \in \mathbb{R}^D$ be the feature importance score vector for the testing instance pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Based on Theorem 1, we know that $\sum_{d=1}^{D} \phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) = \zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Then, we can calculate the average feature importance score over all the features as $\bar{\phi} = \zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)/D$. Suppose $\tilde{\boldsymbol{\phi}}^* \in \mathbb{R}^D$ denotes the deviation vector, in which the d-th element $\tilde{\phi}_d^*$ is calculated as $\tilde{\phi}_d^* = \phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) - \bar{\phi}$. Obviously, we can get $\boldsymbol{\phi} = \bar{\phi}\boldsymbol{I}_D + \tilde{\boldsymbol{\phi}}^*$, where $\boldsymbol{I}_D$ is a D dimensional vector with all entries being one. For the non-discriminative features, their importance scores are close to the average value $\bar{\phi}$, and the deviation values of them are close to zero. In contrast, for the discriminative features, their importance scores deviate from the average value $\bar{\phi}$ far away.

Since the average value $\bar{\phi}$ can be easily calculated, to reduce the computational complexity, our goal here is to first optimize the deviation vector $\tilde{\boldsymbol{\phi}}^*$, and then derive the feature importance score vector based on $\boldsymbol{\phi} = \bar{\phi}\boldsymbol{I}_D + \tilde{\boldsymbol{\phi}}^*$. Considering that only a subset of features contain the discriminative information, we assume that $\tilde{\boldsymbol{\phi}}^*$ is s-sparse, and $\tilde{\boldsymbol{\phi}}^*$ is called s-sparse if $\|\tilde{\boldsymbol{\phi}}^*\|_{\mathcal{L}_0} \leq s$, where $\|\tilde{\boldsymbol{\phi}}^*\|_{\mathcal{L}_0} = \lim_{p\to 0} \sum_{d=1}^{D} |\tilde{\phi}_d^*|^p = \sum_{d=1}^{D} \mathbb{I}(\tilde{\phi}_d^* \neq 0)$. Specifically, the quantity $\|\tilde{\boldsymbol{\phi}}^*\|_{\mathcal{L}_0}$ computes the number of nonzero elements in the feature importance score vector $\tilde{\boldsymbol{\phi}}^*$. To calculate the sparse vector $\tilde{\boldsymbol{\phi}}^*$, we propose to solve the following optimization problem

$$\tilde{\boldsymbol{\phi}}^* = \text{argmin}_{\tilde{\boldsymbol{\phi}} \in \mathbb{R}^D} \{\|\tilde{\boldsymbol{\phi}}\|_{\mathcal{L}_0} : \text{ s.t. } \|\boldsymbol{b} - \boldsymbol{A}(\bar{\phi}\boldsymbol{I}_D + \tilde{\boldsymbol{\phi}})\|_{\mathcal{L}_2} = 0\},$$

where $\boldsymbol{b} \in \mathbb{R}^M$ is an observed measurement vector, and $\boldsymbol{A} \in \mathbb{R}^{M \times D}$ is a random Bernoulli measurement matrix for which the number of rows is far less than that of columns (i.e., $M \ll D$). The entries of $\boldsymbol{A}$ take the value $\frac{1}{\sqrt{M}}$ or $-\frac{1}{\sqrt{M}}$ with equal probability. In the above equation, we aim to use much fewer measurements to calculate $\tilde{\boldsymbol{\phi}}^*$ and further derive the feature importance score vector $\boldsymbol{\phi}$.

However, the above $\mathcal{L}_0$-norm minimization formulation is NP-hard because it involves enumerative search and is computationally intractable for practical applications. Besides scalability, another important requirement for real-world applications is the robustness to noise, namely, the observation vector $\boldsymbol{b}$ may be corrupted by data noise. Without loss of generality, we assume that the measurement vector $\boldsymbol{b}$ is corrupted by noise of magnitude up to ϵ. To address the computationally intractable problem, we propose to use the convex relaxation by replacing the $\mathcal{L}_0$-norm with the $\mathcal{L}_1$-norm. To take the noise into account, we propose to relax the equality constraint as $\|\boldsymbol{b} - \boldsymbol{A}(\bar{\phi}\boldsymbol{I}_D + \tilde{\boldsymbol{\phi}})\|_{\mathcal{L}_2} \leq \epsilon$. Then, we can derive the following optimization problem

$$\hat{\boldsymbol{\phi}}^* = \text{argmin}_{\hat{\boldsymbol{\phi}} \in \mathbb{R}^D} \{\|\hat{\boldsymbol{\phi}}\|_{\mathcal{L}_1} : \text{ s.t. } \|\boldsymbol{b} - \boldsymbol{A}(\bar{\phi}\boldsymbol{I}_D + \hat{\boldsymbol{\phi}})\|_{\mathcal{L}_2} \leq \epsilon\},$$

where $\epsilon > 0$ is a pre-defined noise level. Finally, $\tilde{\boldsymbol{\phi}}^* \in \mathbb{R}^D$ can be well approximated by $\hat{\boldsymbol{\phi}}^*$ based on Eqn. (2.8). Note that the above problem is an underdetermined linear problem since $M \ll D$, and the $\mathcal{L}_1$-norm minimization solution is also the sparsest possible solution [19, 20]. The problem can be recast as a linear program and can be solved by conventional methods such as interior-point methods. However, these methods suffer from poor scalability for real-world problems with large-scale data. To address this challenge, we propose to use the fast iterative shrinkage-threshold method [21] to solve the above optimization problem, and the proposed RAIPair is summarized in Algorithm 1. In this algorithm, we first estimate the measurement vector $\boldsymbol{b}$ from a set of random permutations (i.e., $\{\mathcal{O}_h\}_{h=1}^H$) (Step 1-12), and then derive the approximated importance score vector $\tilde{\phi}$ by solving the $\mathcal{L}_1$ minimization problem (Step 13-14). In Theorem 2, we also present the approximation error bound for the proposed RAIPair. The computational complexity of RAIPair on all features' contributions is $H * D = O(\log(\log D) * D)$, where D is the feature dimension. Thus, the computational complexity of RAIPair is much lower than that of ASIPair

Algorithm 1 The robust approximation interpretation method for pairwise models

Input: Pairwise model ζ, the number of measurements M, the test pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$, the number of permutations H, and the random Bernoulli matrix $\boldsymbol{A}$.

1: **for** h $\leftarrow 1$ to H **do**
2: Randomly select the permutation $\mathcal{O}_h \in \pi(D)$;
3: **for** $d \leftarrow 1$ to D **do**
4: $\Delta_d^h(\boldsymbol{x}_i, \boldsymbol{x}_j, P^d_{\mathcal{O}_h}, \zeta) = \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|x_i^{P^d_{\mathcal{O}_h}\cup\{d\}}, \boldsymbol{x}_j^{P^d_{\mathcal{O}_h}\cup\{d\}})] - \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P^d_{\mathcal{O}_h}}, \boldsymbol{x}_j^{P^d_{\mathcal{O}_h}})]$;
5: **end for**
6: **for** $m \leftarrow 1$ to M **do**
7: $Y_{m,h} \leftarrow \sum_{d=1}^D \boldsymbol{A}_{m,d}\Delta_d^h(\boldsymbol{x}_i, \boldsymbol{x}_j, P^d_{\mathcal{O}_h}, \zeta)$;
8: **end for**
9: **end for**
10: **for** $m \leftarrow 1$ to M **do**
11: $b_m = \frac{1}{H}\sum_{h=1}^H Y_{m,h}$; // b_m is the m-th element in $\boldsymbol{b}$
12: **end for**
13: $\hat{\boldsymbol{\phi}}^* = \text{argmin}_{\hat{\boldsymbol{\phi}}\in\mathbb{R}^D}\{\|\hat{\boldsymbol{\phi}}\|_{\mathcal{L}_1} : \text{ s.t. } \|\boldsymbol{b} - \boldsymbol{A}(\bar{\phi}\boldsymbol{I}_D + \hat{\boldsymbol{\phi}})\|_{\mathcal{L}_2} \leq \epsilon\}$;
14: **return** the approximated feature importance score vector $\tilde{\boldsymbol{\phi}} = \bar{\boldsymbol{\phi}}\boldsymbol{I}_D + \hat{\boldsymbol{\phi}}^*$.

Theorem 2. *Assume that the range of the predictions made by the pairwise model $\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is $[-r, +r]$, and the restricted isometry constant δ_{2s} of the matrix $\boldsymbol{A} \in \mathbb{R}^{M \times D}$ satisfies $\delta_{2s} < \frac{3}{4+\sqrt{6}} \approx 0.465$. Let $\sigma_s(\boldsymbol{\phi})_{\mathcal{L}_1} := \inf\{\|\boldsymbol{\phi} - \Psi\|_{\mathcal{L}_1}, \Psi \text{ is } s\text{-sparse}\}$, $\epsilon > 0$, $0 < \delta < 1$, and C be a universal constant. Then, if $M \geq C(0.465)^{-2}(2s\log(D/(2s)) + \log(2/\delta))$ and $\frac{2r^2}{\epsilon^2}\log\frac{4M}{\delta} \leq H$, we then can derive*

$$\|\tilde{\boldsymbol{\phi}} - \boldsymbol{\phi}\|_{\mathcal{L}_2} = \|\hat{\boldsymbol{\phi}}^* - \hat{\boldsymbol{\phi}}\|_{\mathcal{L}_2} \leq \Phi_1\epsilon + \Phi_2\frac{\sigma_s(\boldsymbol{\phi})_{\mathcal{L}_1}}{\sqrt{s}}, \tag{2.8}$$

where ϵ denotes the noise amount, $\Phi_1 \in \mathbb{R}$ and $\Phi_2 \in \mathbb{R}$ are two constants that only depend on δ_{2s}. Note that H denotes the number of random permutations used to estimate the measurement vector $\boldsymbol{b}$.

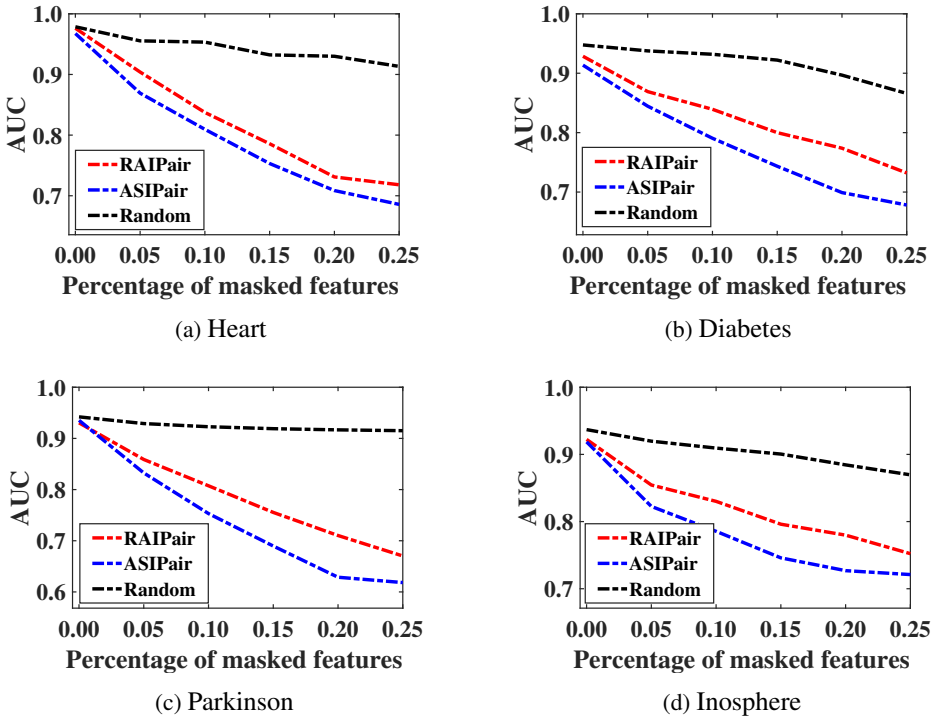
Based on Theorem 2, we can bound the error between the proposed ASIPair and its approximated version (i.e., RAIPair). In fact, both ASIPair and RAIPair use the same strategy to calculate the marginal function Δ_d, which can be seen in Eqn.(2.1) and Step 4 of Algorithm 1. The difference between ASIPair and RAIPair is that RAIPair uses very few operations to average Δ_d to approximate ASIPair.

2.3 Experiments

We conduct experiments on both real-world and synthetic datasets to evaluate the performance of the proposed interpretation methods. All the experiments are conducted 10 times and we report the average results.

Datasets

For real-world datasets, we adopt four UCI datasets (i.e., Heart, Diabetes, Parkinson and Ionosphere), and the MNIST 1V9 dataset [22] that is a subset of the 784-dimensional MNIST set. The statistical information of these real-word datasets is described in Table 2.1. For the synthetic dataset, we use the following method to generate the data: We first generate N instances $\{\boldsymbol{x}_i\}_{i=1}^N$, where $\boldsymbol{x}_i$ is a D-dimensional feature vector in which each element is randomly generated in range $(-1, 1)$. Then we build a linear classifier with the weight vector $\boldsymbol{w}$ in which each element $w_i \sim U(-0.5, 0.5)$. Finally, we use the linear classifier to generate the label of each instance. For each dataset, we randomly select 80% of the instances as the training set to train the pairwise model, and take the rest instances as the test set.



(a) Heart (b) Diabetes

(c) Parkinson (d) Inosphere

Figure 2.1: The AUC metric w.r.t the percentage of masked features.

Performance Measure

We evaluate the performance of the proposed interpretation methods through observing the change of the predicted results after masking a proportion of the top features ranked by the learned feature importance scores. Specifically, given a trained pairwise model and a test instance pair, both the proposed ASIPair and RAIPair can generate a vector of importance scores that reflects all features' contributions to the pairwise prediction on this test pair. When evaluating the performance of each proposed method, we first rank the importance scores and mask a proportion of the top ranked features. Then, we measure the change of the result predicted by the pairwise model before and after masking the features. The larger the predicted result changes, the more important the masked features are. In addition, considering that there is no existing interpretation method designed for pairwise learning, we adopt the random masking method as the baseline, in which we randomly select a proportion of features and then mask them.

Table 2.1: The statistics of the datasets.

Dataset	Size	Dimension
Heart	303	23
Diabetes	768	9
Parkinson	195	22
Ionosphere	351	34
MNIST	2,134	784

Interpretation for AUC Maximization

We first study the performance of the proposed interpretation methods on a widely used AUC maximization model, i.e., OPAUC [23], which aims to maximize the AUC metric by going through the training data only once without storing the entire training dataset. Here we evaluate the performance of the proposed methods through observing the change of AUC metric before and after masking the top features ranked by the calculated importance scores. The AUC metric of a pairwise model is equal to the probability that the model ranks a randomly chosen positive instance higher than a randomly chosen negative instance [24]. The lower the AUC metric, the larger the change of the predicted results, the better the proposed interpretation method. In this experiment, we vary the percentage of masked features over the total number of features from 0.001 to 0.25, and the AUC metric on the four UCI datasets are shown in Figure 2.1. As we can see, compared with masking a proportion of randomly selected features, masking the top ranked features derived based on our proposed interpretation methods have more effect on the predicted results, which means both ASIPair and RAIPair can effectively identify important features that make key contributions to the predicted results. Additionally, this figure also shows that the performance of ASIPair is little better than that of RAIPair.

Efficiency

We also evaluate the efficiency of the proposed interpretation methods. In this experiment, we adopt a widely used metric learning model, i.e., LowRank [25], which aims to learn a metric that can measure the similarity degree between a pair of instances. In this experiment, we generate several synthetic datasets by varying the value of D from 2 to 9. The size of each synthetic dataset (i.e., N) is set as 1000. We then evaluate the running time of ASIPair and RAIPair on each dataset, and the average result on all testing instance pairs is shown in Figure 2.2. From

this figure, we can see that the running time of RAIPair is polynomial with respect to the input feature dimension D while that of ASIPair is approximately exponential with respect to D. When the number of features increases, RAIPair shows great advantage in running time, which verifies our conclusion that RAIPair is much more efficient than ASIPair.

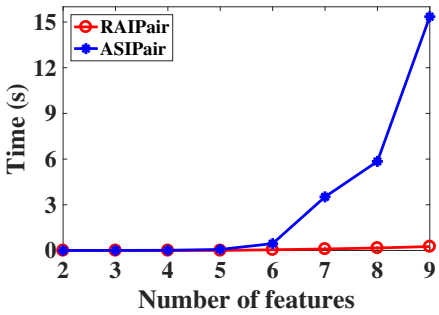


Figure 2.2: Running time of ASIPair and RAIPair.

Sparsity

In addition, we evaluate the performance of the proposed RAIPair on sparse feature selection using the MNIST dataset. The pairwise model to be interpreted is OPAUC. We randomly select four testing instance pairs and report the deviation score of each feature (i.e., $\hat{\phi}_d^* \in \hat{\boldsymbol{\phi}}^*$) calculated by RAIPair in Figure 2.3. The results in this figure show that RAIPair can effectively select a subset of features that make key contributions to the predicted result. Take Figure 2.3a as an example. We can see the deviation scores of most features are close to 0 and only a subset of features have large deviation scores, which means the importance scores of most features are around the average value (i.e., $\bar{\phi}$) and only a subset of features play an important role in providing discriminative information for the final predicted result. Based on Figure 2.3a, we can also know only a subset of features contain discriminative information.

Visualization

Last but not least, we provide some visualization results to further evaluate the effectiveness of the proposed methods. In this experiment, we use RAIPair to interpret the predictions made by the AUC maximization model OPAUC on the MNIST 1V9 dataset. The results for one testing instance pair (a picture for digital 9 and a picture for digital 1) are shown in Figure 2.4. Figure 2.4a shows the correctly classified testing instance pair, which means that the positive instance (the digit 9) ranks higher than the negative instance (the digit 1). After applying RAIPair

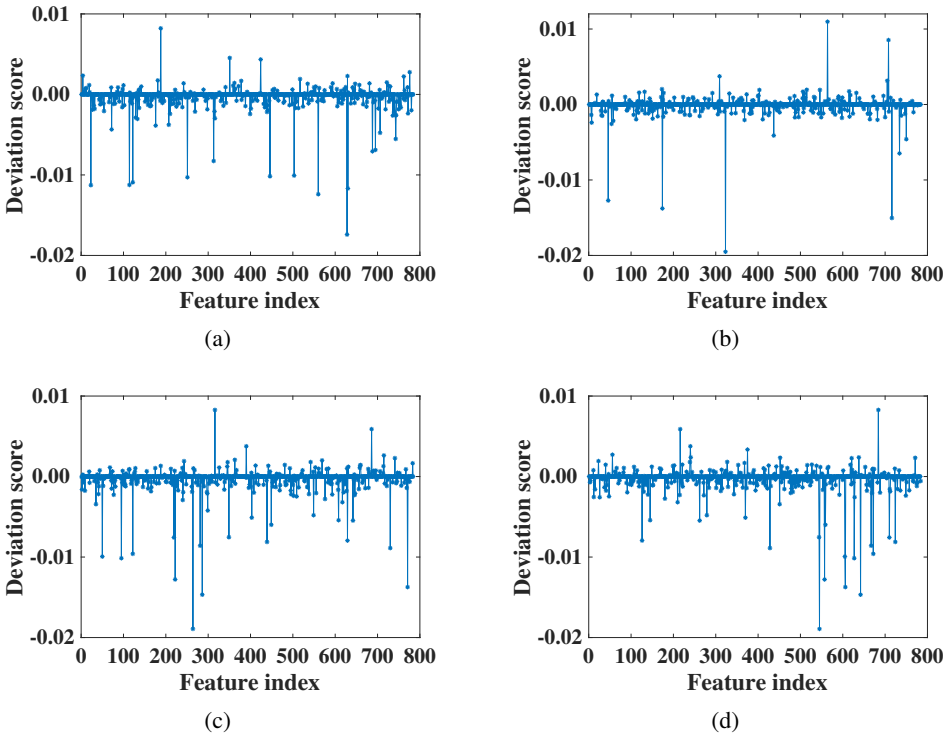


Figure 2.3: The calculated deviation scores for testing instance pairs on MNIST dataset. The results in (a)-(d) are for four different instance pairs.

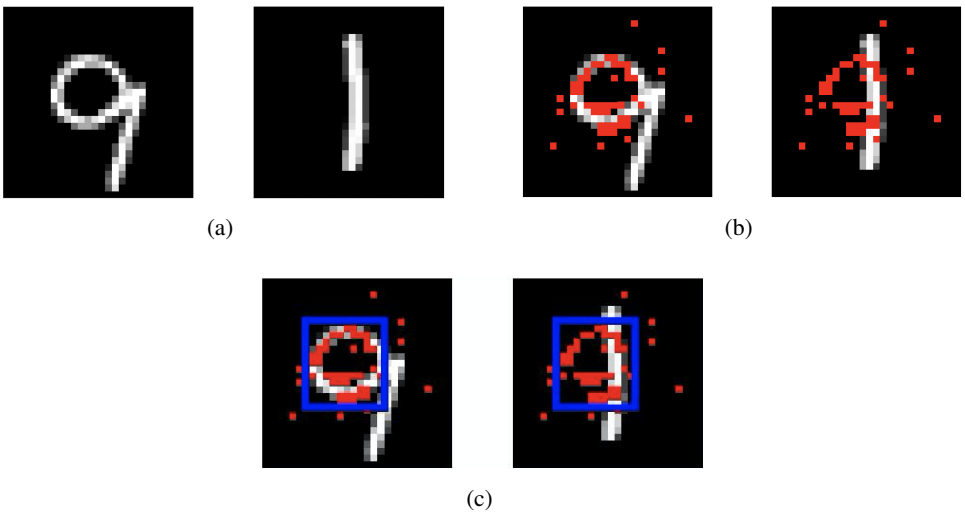


Figure 2.4: Visualization results generated by the proposed RAIPair on the MNIST dataset.

to this pair, we can get the importance score vector associated with the underlying features of this pair. We then rank these importance scores and select the 8% of the top-ranked features, which are highlighted with red color in Figure 2.4b. In Figure 2.4c, we use blue rectangles to highlight the salient parts of the selected features that make key contributions to the pairwise prediction. As we can see, the proposed RAIPair can accurately capture the salient parts of the input features for the pairwise predicted result, and these parts agree well with the empirical intuition of humans.

2.4 Related Work

Although pairwise learning has been well studied in many works [4, 5], there is no existing work that considers how to interpret the predictions made by the learned pairwise models. Recently, a wide variety of interpretation methods have been developed to provide explanations for the pointwise learning models (e.g., classification model) through scoring the importance of each input feature for a given instance [10, 11, 12, 13, 14, 15, 16, 17, 26, 27]. Among these pointwise interpretation methods, the Shapley-value-based methods have drawn significant attention as they can provide strong theoretical foundations and better agree with the human intuition [13, 14, 15, 16, 17, 28, 29]. However, these pointwise interpretation methods do not take into account pairwise input and cannot be directly used in pairwise learning. To calculate the importance scores of the input features, these methods usually need a pre-defined reference vector and require all the instances to use the same reference vector, which is unreasonable for pairwise models. Furthermore, these methods usually assume that the input features do not correlate with each other. However, in practice, the features of an instance may correlate with each other and the correlations can also affect the predicted results. Thus, we take into account feature correlations when we design the interpretation methods for pairwise models.

Considering that Shapley value approach is usually computationally challenging, existing works also take measures to reduce the computational cost. Specifically, the authors in [17, 14, 13] propose various sampling-based approximation methods. However, these sampling-based methods may suffer from high variance when the number of instances to be collected is limited. The authors in [15] develop two approximation methods based on the assumption that the input features have an underlying graph structure. By assuming that the hidden layer is distributed with an isotropic Gaussian, the authors in [16] propose an approximation method for deep neural networks. Different from these methods, our proposed approximation method RAIPair is more general and does not make any assumptions on the input data structure.

In addition, this work is significantly different from existing pairwise feature selection methods [30, 31] for pairwise learning. Existing pairwise feature selection methods aim to alter the training phase to learn a subset of features that are relevant to the targeted model. Also, even for these selected features, they cannot distinguish their relative relevance scores. However, our proposed interpretation methods only involve the testing stage, and aim to interpret each individual pairwise prediction that is made by the trained pairwise model.

2.5 Conclusions

In this chapter, we investigate how to enable interpretation in pairwise learning. Specifically, we first propose a novel adaptive interpretation method for pairwise learning (i.e., ASIPair), based on which a vector of importance scores associated with the underlying features of a testing instance pair can be adaptively calculated, and these scores can be used to indicate which features make key contributions to the final prediction. Considering that the proposed ASIPair is computationally challenging, we further propose a novel robust approximation interpretation method for pairwise models (i.e., RAIPair). This method is not only much more efficient but also robust to data noise. Theoretical analysis and extensive experiments demonstrate the effectiveness of the proposed interpretation methods for pairwise learning.

Chapter 3

Understanding Generalization in Deep Metric Learning

3.1 Introduction

Measuring the similarity between data samples plays an important role in many machine learning and data mining algorithms. Although some simple metrics (e.g., Euclidean distances) can be used to measure the similarity between samples, they have no capability to capture the statistical regularities in the data, and thus largely degrade the performance of the algorithms [32]. To address this challenge, metric learning, whose goal is to learn a distance metric that can capture the important relationships among data samples, has drawn significant attention [33, 32, 34, 35, 36, 8, 7]. The basic idea of most metric learning methods is first to learn a Mahalanobis distance metric, which is a linear mapping to project the original samples into a new feature space, and then determine the similarity of samples in the new feature space. However, these conventional Mahalanobis-based methods usually have inherent limits on their mapping capability, and thus fail to achieve good performance when handling data with nonlinear structures.

Given that deep learning has good capability of modeling the nonlinearity of samples, there has been significant effort [37, 38, 39, 40, 41, 42] studying deep metric learning (DML), which unifies deep learning and metric learning into a joint learning framework. The key idea of DML is to explicitly train a deep neural network and derive a set of hierarchical nonlinear mappings, based on which the data samples can be projected into a new feature space for comparing. The

derived nonlinear mappings are capable of guaranteeing that the distance between similar samples is close and the distance between dissimilar samples is far in the new feature space [42]. Additionally, compared with the traditional metric learning methods, DML has shown better scaling properties when handling massive data. Although DML has achieved practical success in many applications, there is no existing work that theoretically analyzes the generalization error of DML, which is the difference between the empirical and expected errors, and it can measure how good a learned model is able to perform on unseen data. A comprehensive theoretical generalization analysis is essential for DML as it can not only provide much important information about the practical performance of DML but also guide the design of effective network architectures for DML.

In this chapter, we try to fill up this research gap and derive the generalization bound for DML. Here we consider a general case where the DML models adopt the dropout strategies, in which each connection is kept in the neural network with a predefined retention rate during the training process. In particular, when these retention rates are set to ones, it reduces to the generalization bound of the DML model without dropout. Based on the derived generalization bound, we can have a good understanding of the generalization properties of DML in many applications, especially in the settings where dropout is used to train DML models with the goal of achieving good generalization performance [43]. In practice, specifying these predefined retention rates for dropout is usually difficult as it requires significant levels of experience and domain knowledge. However, the derived generalization error bound for DML can be treated as a function related to the weight parameters of the neural networks and the retention rates for dropout. Based on this fact, we propose a novel Adaptive Dropout based DML method (**ADroDML**) by incorporating the obtained generalization bound to the objective function of DML models as a regularizer. The goal of incorporating the bound-based regularizer is to reduce the model complexity for DML to give a lower error on future unseen data. ADroDML allows us to jointly learn the weight parameters and the retention rates for DML in a theoretically justified way, and it can achieve better performance compared with existing DML works that require predefined retention rates. Extensive experiments on real-world datasets verify the findings derived from the error bound and show the effectiveness of the proposed ADroDML.

3.2 Preliminary

In this section, we introduce the DML model that takes dropout into account. Without loss of generality, we use the widely adopted Siamese network [37, 38, 44] as our example. Suppose

$\boldsymbol{z} = \{\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^n$ denote the labeled training dataset, where $\boldsymbol{x}_i \in \mathbb{R}^d$ is a d dimensional feature vector and $y_i \in \{-1, 1\}$ is the class label. DML aims to train a L-layer neural network to predict whether two input samples (i.e., two feature vectors) are similar or not. Assume that the trained L-layer neural network is parameterized by the weights $\boldsymbol{W} = \{\boldsymbol{W}^l \in \mathbb{R}^{h_l * h_{l-1}}\}_{l=1}^L$ (note that the biases are included in $\boldsymbol{W}$ with a corresponding fixed input of 1 for simplicity), where h_l represents the number of neurons in the l-th layer of the network and $h_0 = d$. We denote the retention rates for dropout as $\boldsymbol{\rho} = \{\rho^l\}_{l=1}^L$, where ρ^l represents the retention rate for the l-th layer. Then, given the input sample $\boldsymbol{x}_i \in \mathbb{R}^d$, the output of the l-th layer in the network can be written as

$$\begin{aligned} f^l(\boldsymbol{x}_i; \boldsymbol{W}^{1:l}, \boldsymbol{M}^{1:l}) &= (\boldsymbol{W}^l \odot \boldsymbol{M}^l)\sigma(f^{l-1}(\boldsymbol{x}_i, \boldsymbol{W}^{1:l-1}, \boldsymbol{M}^{1:l-1})) \\ &= (\boldsymbol{W}^l \odot \boldsymbol{M}^l)\sigma((\boldsymbol{W}^{l-1} \odot \boldsymbol{M}^{l-1})\sigma(\cdots\sigma((\boldsymbol{W}^1 \odot \boldsymbol{M}^1)\boldsymbol{x}_i))), \end{aligned}$$

where $\sigma(\cdot)$ denotes the activation function, $\boldsymbol{M}^{1:l} = \{\boldsymbol{M}^1, \boldsymbol{M}^2, ..., \boldsymbol{M}^l\}$, and $\boldsymbol{M}^l = \{M_{ij}^l\}_{i=1,j=1}^{h_l, h_{l-1}} \in \mathbb{R}^{h_l * h_{l-1}}$ is a binary matrix where each element $M_{ij}^l \in \{0, 1\}$ is drawn from the distribution $Bern(\rho^l)$. The term $(\boldsymbol{W}^l \odot \boldsymbol{M}^l)$ corresponds to dropping each of the weight parameters $\boldsymbol{W}^l = \{W_{ij}^l\}_{i=1,j=1}^{h_l, h_{l-1}}$ independently with probability $1 - \rho^l$. In particular, $f^1(\boldsymbol{x}_i, \boldsymbol{W}^1, \boldsymbol{M}^1) = (\boldsymbol{W}^1 \odot \boldsymbol{M}^1)\boldsymbol{x}_i$. Note that the most top level representation of the input $\boldsymbol{x}_i$, i.e.,$f^L(\boldsymbol{x}_i; \boldsymbol{W}^{1:L}, \boldsymbol{M}^{1:L})$, is a random vector due to the adopted Bernoulli random variable $\boldsymbol{M}$. Thus, following [45, 46, 47], we use the expected value $f^L(\boldsymbol{x}_i; \boldsymbol{W}, \boldsymbol{\rho}) = E_{\boldsymbol{M}}[f^L(\boldsymbol{x}_i; \boldsymbol{W}, \boldsymbol{M})]$ as the deterministic output of the neural network with dropout.

Specifically, DML aims to seek the nonlinear embedding function $f^L : \mathbb{R}^d \to \mathbb{R}^{h_L}$, which guarantees that the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is smaller than a pre-specified margin $\gamma > 0$ in the transformed space if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar, and larger than γ in the transformed space if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are dissimilar. To learn a good embedding function f^L with such desirable properties, the widely adopted method is to minimize the following empirical risk with dropout over the given training samples

$$\mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W}) = \frac{2}{n(n-1)} \sum_{i<j} g(1 + y_{ij}(D(f^L(\boldsymbol{x}_i; \boldsymbol{W}, \boldsymbol{\rho}), f^L(\boldsymbol{x}_j; \boldsymbol{W}, \boldsymbol{\rho})) - \gamma)), \tag{3.1}$$

where $y_{ij} = y_i y_j \in \{-1, 1\}$ is the similarity label, γ is the unit margin and $g(\cdot)$ is the hinge

loss. $D(f^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho}), f^L(\boldsymbol{x}_j; \boldsymbol{W}, \boldsymbol{\rho}))$ is defined as follows:

$$\begin{aligned} &D(f^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho}), f^L(\boldsymbol{x}_j; \boldsymbol{W}, \boldsymbol{\rho})) \\ &= (f^L(\boldsymbol{x}_i; \boldsymbol{W}, \boldsymbol{\rho}) - f^L(\boldsymbol{x}_j; \boldsymbol{W}, \boldsymbol{\rho}))^T (f^L(\boldsymbol{x}_i; \boldsymbol{W}, \boldsymbol{\rho}) \\ &- f^L(\boldsymbol{x}_j; \boldsymbol{W}, \boldsymbol{\rho})) = \sum_{k=1}^{h_L} (f_k^L(\boldsymbol{x}_i; \boldsymbol{W}, \boldsymbol{\rho}) - f_k^L(\boldsymbol{x}_j; \boldsymbol{W}, \boldsymbol{\rho}))^2, \end{aligned} \tag{3.2}$$

where $f_k^L(\cdot)$ represents the output of the k-th neuron in the L-th layer of the network. In practice, the deterministic output $f^L(\boldsymbol{x}_i; \boldsymbol{W}, \boldsymbol{\rho})$ can be derived by introducing a deterministic scaling factor (i.e., $E[\boldsymbol{M}^l]$) for each layer to replace the random dropout variable [46, 48]. Then, the empirical loss $\mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})$ can be easily computed since it only involves a single deterministic network. Note that $\mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})$ is also known as the contrastive loss and can measure how well f^L is able to place similar samples nearby and keep dissimilar samples separated.

3.3 Generalization Analysis for Deep Metric Learning

In this section, we derive the generalization bound for DML, which is the difference between the expected and empirical risks. We derive the generalization bound by analyzing

$$\text{the generalization bound} \triangleq \mathcal{R}(\boldsymbol{W}_{\boldsymbol{z}}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W}_{\boldsymbol{z}}), \tag{3.3}$$

where $\boldsymbol{W}_{\boldsymbol{z}} = \arg\min_{\boldsymbol{W}} \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})$ is the empirical risk minimizer and $\mathcal{R}(\boldsymbol{W}_{\boldsymbol{z}}) = E_{\boldsymbol{z}}[\mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W}_{\boldsymbol{z}})]$ represents the expected risk of the trained model on the whole space of possible data. Note that the empirical risk $\mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W}_{\boldsymbol{z}})$ is also known as a U-statistic [49] in the statistic literature, and is no longer an empirical average of independent random samples from $\boldsymbol{z}$ as in the standard deep learning setting, but rather an average of *pairs* of random samples from $\boldsymbol{z}$. Thus, it is more challenging to perform generalization analysis for DML. To address this challenge, we develop a novel analysis method by extending Rademacher complexity analysis [50] to the setting of DML, and then we derive Theorem 1 that describes the generalization bound for DML with dropout.

Theorem 3. ***(Generalization Bound for DML with Dropout)*** *Suppose the deep neural network needed to be learned has L layers and the weight parameter $\boldsymbol{W}^l$ ($l \in [L]$) satisfies $\|\boldsymbol{W}^l\|_F \leq B^l$. Let $\sigma(\cdot)$ be a 1-Lipschitz activation function (e.g., ReLU) and $\boldsymbol{\mathcal{X}} \in [0,1]^d$ denote the feature space. Then for any $\delta \in (0,1)$, with probability $1-\delta$ we have*

$$\mathcal{R}(\boldsymbol{W_z}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W_z}) \leq 3V_1\sqrt{\frac{2\log(1/\delta)}{n}} + 6\sqrt{1/\lfloor\frac{n}{2}\rfloor} \tag{3.4}$$
$$+ 32h_L B^L V^L(\sqrt{2L\log 2}+1)(\prod_{l=1}^{L} B^l)(\prod_{l=1}^{L}\sqrt{\rho^l})\sqrt{d/\lfloor\frac{n}{2}\rfloor},$$

where $|g(\cdot)| \leq V_1$*, and* $\|\sigma(f^{L-1}(\cdot))\| \leq V^L$*. Note that* $g(\cdot)$ *denotes the loss function, and* $f^{L-1}(\cdot)$ *represents the output of the* $(L-1)$*-th layer of the learned neural network.*

Proof. The detailed proof for this theorem is provided in Appendix. □

Observations. With Theorem 1, we can derive the following observations, which can help explain the behaviors of existing DML models and guide the design of good neural networks for DML.

- The generalization bound (i.e., $\mathcal{R}(\boldsymbol{W_z})-\mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W_z})$) decreases monotonically at the rate of $\mathcal{O}(\sqrt{1/n})$ when the training data size n increases. In particular, $(\mathcal{R}(\boldsymbol{W_z})-\mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W_z})) \to 0$ when $n \to +\infty$, which indicates that the DML models can achieve good generalization performance when the training data size is sufficiently large.

- The generalization bound is also related to the dimensionality of the feature vector (i.e., d). In the cases where the value of d is extremely small (e.g., $d = 0$), we may get a small generalization bound. However, the empirical loss may be very large since the learned model cannot capture the particular characteristics of the data [46]. On the other hand, high-dimensional input features (i.e., d is extremely large) usually contain much noisy information, which can hide the relationship between the learning task and the most relevant features [51] and thus incurs large generalization bounds. Thus, a proper feature set that dominates the underlying learning task should be selected, and are then fed into the network.

- The magnitude of the weight parameters (i.e., $\|\boldsymbol{W}^l\|_F$) at the end of the learning process is critical to the generalization performance, and small magnitude of the weights is preferred. By observing this, explicit regularizers can be imposed on the weight parameters, which is achieved by penalizing the norm of the optimal solution. In this way, the generalization bound can be dramatically reduced when the magnitude of the weight parameters are very large. Also, weight-decay can be adopted to avoid choosing large-magnitude weights, which can improve the generalization performance.

- The multiplicative term $\prod_{l=1}^{L}\sqrt{\rho^l}$ which is related to the retention rates of dropout helps us to understand how the dropout method works. When $\rho^l = 0$, the above bound is tight since the features from the training samples have no influence on the output [46]. When $\forall l \in [L]$, $\rho^l = 1$, it reduces to the complexity of a standard model. That is to say, when these retention rates are set to ones, it reduces to the generalization bound of the DML model without dropout. For other cases where $\rho_l \in (0, 1)$, we can obtain that $\prod_{l=1}^{L}\sqrt{\rho^l} < 1$, which means that dropout can reduce the generalization bound. However, when continuously decreasing the retention rates for dropout, the quality of the learned model may deteriorate [46]. The reason is that the learned model may be tuned to the particular training samples, rather than the underlying characteristics of the data. Thus, specifying optimal retention rates for dropout is very crucial in practice.

- The generalization bound is also affected by V^L, which denotes the bounded output range of the activation function $\sigma(\cdot)$ on the $(L-1)$-th layer. This theorem implies that batch normalization can improve the generalization performance as it is an operator that normalizes the output of the previous layer within each mini-batch, especially in the settings where the output range is extremely large.

Two other factors that affect the derived generalization bound are L and h_L, which can provide suggestions on non-extreme-deep neural networks and non-extreme-wide output layers, respectively. As we can see, all the above observations are consistent with the widely used network architectures in practice. Additionally, Theorem 1 can be easily generalized to other situations where different loss functions (e.g., the triplet network-based DML models), activation functions and types of dropout (e.g., dropout of both weights and units) are adopted.

3.4 Adaptive Dropout for Deep Metric Learning

The above generalization analysis implies that taking dropout into account during the training process can help to reduce the generalization error of DML. However, the retention rates for dropout are usually pre-defined based on the experience and domain knowledge, and fixed throughout the training process. It is still not clear how to choose the optimal retention rates such that the learned DML model can achieve the best performance. To address this challenge, in this section, we propose a novel adaptive dropout based DML method (called **ADroDML**) by incorporating the derived generalization error bound into the objective function of DML as a regularizer, and let this error bound guide the choice of the retention rates. Specifically, the

retention rates $\boldsymbol{\rho} = \{\rho^l\}_{l=1}^L$ for dropout and the weight parameters $\boldsymbol{W} = \{\boldsymbol{W}^l\}_{l=1}^L$ of the network are unified into one objective function, which is defined as

$$\min_{\boldsymbol{W},\boldsymbol{\rho}} \mathcal{L}(\boldsymbol{W},\boldsymbol{\rho}) = \frac{2}{n(n-1)} \sum_{i<j} g(1 + y_{ij}(D(f^L(\boldsymbol{x}_i,\boldsymbol{W},\boldsymbol{\rho}), f^L(\boldsymbol{x}_j,\boldsymbol{W},\boldsymbol{\rho})) - \gamma)) + \beta\Omega(\boldsymbol{W},\boldsymbol{\rho}), \tag{3.5}$$

where $\beta > 0$ is the associated regularization parameter, and the regularization term $\Omega(\boldsymbol{W},\boldsymbol{\rho})$ is derived from the generalization bound given in Eqn. (3.4). There are two terms in the right hand side of Eqn. (3.5). The first term is used to penalize the large distance between similar sample pairs and penalize the small distance between dissimilar instance pairs. The goal of the second term is to reduce the model complexity for DML to give lower error on future unseen data. The term $\Omega(\boldsymbol{W},\boldsymbol{\rho})$ in Eqn. (3.5) is computed as follows

$$\Omega(\boldsymbol{W},\boldsymbol{\rho}) = \Delta * (\prod_{l=1}^{L} \|\boldsymbol{W}^l\|_F)(\prod_{l=1}^{L} \sqrt{\rho^l}),$$

where $\Delta = 32 h_L B^L V^L \sqrt{d/\lfloor \frac{n}{2} \rfloor}(\sqrt{2L\log 2} + 1)$. Note that the first two terms in the right hand of Eqn. (3.4) are omitted since they do not contain either the weight parameters or the retention probability parameters.

Optimization. Next, we discuss how to solve the optimization problem formulated in Eqn. (3.5), where we have two sets of parameters that need to be learned, i.e., $\boldsymbol{W} = \{\boldsymbol{W}^l\}_{l=1}^L$ and $\boldsymbol{\rho} = \{\rho^l\}_{l=1}^L$. Here we solve this optimization problem using the block-coordinate descent algorithm [52], which starts with an initial setting of the parameters, and then optimize $\boldsymbol{W}$ and $\boldsymbol{\rho}$ in an alternating fashion. Specifically, we iteratively conduct the following two steps:

Step 1: Weights update. With an initial estimate of the retention rates $\boldsymbol{\rho} = \{\rho^l\}_{l=1}^L$, we first update the weight parameters $\boldsymbol{W} = \{\boldsymbol{W}^l\}_{l=1}^L$ by minimizing $\mathcal{L}(\boldsymbol{W},\boldsymbol{\rho})$ with fixed $\boldsymbol{\rho} = \{\rho^l\}_{l=1}^L$. By solving this optimization problem, we can then obtain the set of weight parameters $\boldsymbol{W} = \{\boldsymbol{W}^l\}_{l=1}^L$ which minimize $\mathcal{L}(\boldsymbol{W},\boldsymbol{\rho})$ with the fixed retention rates.

Step 2: Retention rates update. In this step, we fix the weight matrices $\boldsymbol{W} = \{\boldsymbol{W}^l\}_{l=1}^L$ for different layers, and then calculate the retention rates $\boldsymbol{\rho} = \{\rho^l\}_{l=1}^L$ through minimizing the objective function $\mathcal{L}(\boldsymbol{W},\boldsymbol{\rho})$ given in Eqn. (3.5).

The above two steps are iteratively conducted until the convergence criterion is satisfied. In this chapter, the convergence criterion is that the difference between the objective function values in two consecutive iterations is less than a threshold. Compared with the dropout strategy that requires to specify the retention rates in advance, the bound-based regularizer enables us

Dataset	Size	Dimension
MNIST 8v9	2,016	28×28
Bone disease	9,704	672
Wine quality	4,898	11

Table 3.1: The statistics of the adopted datasets.

Dataset	# units in the three layers
MNIST 8v9	(784, 64, 10)
Bone disease	(672, 64, 10)
Wine quality	(11, 8, 4)

Table 3.2: The number of units in each layer of the neural networks.

to adaptively optimize the objective and adjust the retention rates for dropout in a theoretically justified way.

3.5 Experiments

3.5.1 Experimental Setup

Datasets. We adopt the following real-world datasets for our experiments: *the MNIST 8v9 dataset*[1] , *the bone disease dataset*[2] , and *the wine quality dataset* [3] . The statistics of these datasets are provided in Table 7.1.

Model settings. Unless otherwise specified, all the neural networks adopted in the experiments have 3 layers. For each dataset, the number of the units in each layer of the neural network is provided in Table 3.2. We implement the DML model using Google Tensorflow, and the training process is done locally using NVIDIA GeForce GTX 1060 GPU. Additionally, Adam optimizer is used in the training process for DML and the learning rate is set as $1e-4$. As for the activation function, we use ReLU because it is a 1-Lipschitz activation function and satisfies the Lipschitz-continuous condition.

[1] http://yann.lecun.com/exdb/mnist/
[2] https://sofonline.epi-ucsf.org/interface/
[3] https://archive.ics.uci.edu/ml/datasets.php

3.5.2 Experiments for Theoretical Observations

We first conduct experiments to verify the derived theoretical observations in Section 3. Specifically, we evaluate the effect of the training data size, batch normalization, regularization, dropout and the input feature dimension on the generalization behavior of DML. Note that the input for DML is a set of sample pairs instead of individual samples. For each dataset, 4,950 sample pairs are selected as the test set (no overlap with training set). Unless otherwise specified, the training data sizes for the MNIST 8v9 dataset, the bone disease dataset and the wine quality dataset are set as 160, 220 and 100, respectively. Correspondingly, the numbers of the generated training sample pairs for the MNIST 8v9 dataset, the bone dataset and the wine quality dataset are 12,720, 24,090 and 4,950, respectively. We do not use the validation set to tune parameters, but assign values by standard settings. When evaluating the effect of the training data size, batch normalization, regularization and dropout, we report the testing loss because the generalization bound is mainly used to measure how well the learned ML model performs on the unseen data (test data). Additionally, we also evaluate the impact of the input feature dimension on the empirical training loss.

The effect of the training data size. To investigate the effect of the training data size (i.e., n) on the generalization behavior of DML, we train the model with different training data sizes and then calculate the testing loss. Here we consider three cases where the training data sizes are set as 20, 50 and 110, respectively. Figure 3.1a and Figure 3.1d show the results on the MNIST 8v9 and bone disease datasets when the number of batches varies. From the two figures, we can see that the larger the training data size, the smaller the testing loss. This verifies that increasing the training data size can potentially improve the generalization performance on unseen data.

The effect of batch normalization. Then we evaluate the effect of batch normalization on the generalization behavior of DML. Here we still adopt the MNIST 8v9 and bone disease datasets. For each dataset, we train the model with and without batch normalization, respectively, and then calculate the testing loss. The results for the two datasets are shown in Figure 3.1b and Figure 3.1e, from which we can see the testing loss of the model trained with batch normalization is lower than that of the model trained without batch normalization. The results verify that batch normalization play an important role to generalize DML.

The effect of regularization. We also evaluate the effect of regularization for DML through explicitly comparing the performance of the DML models with and without regularization.

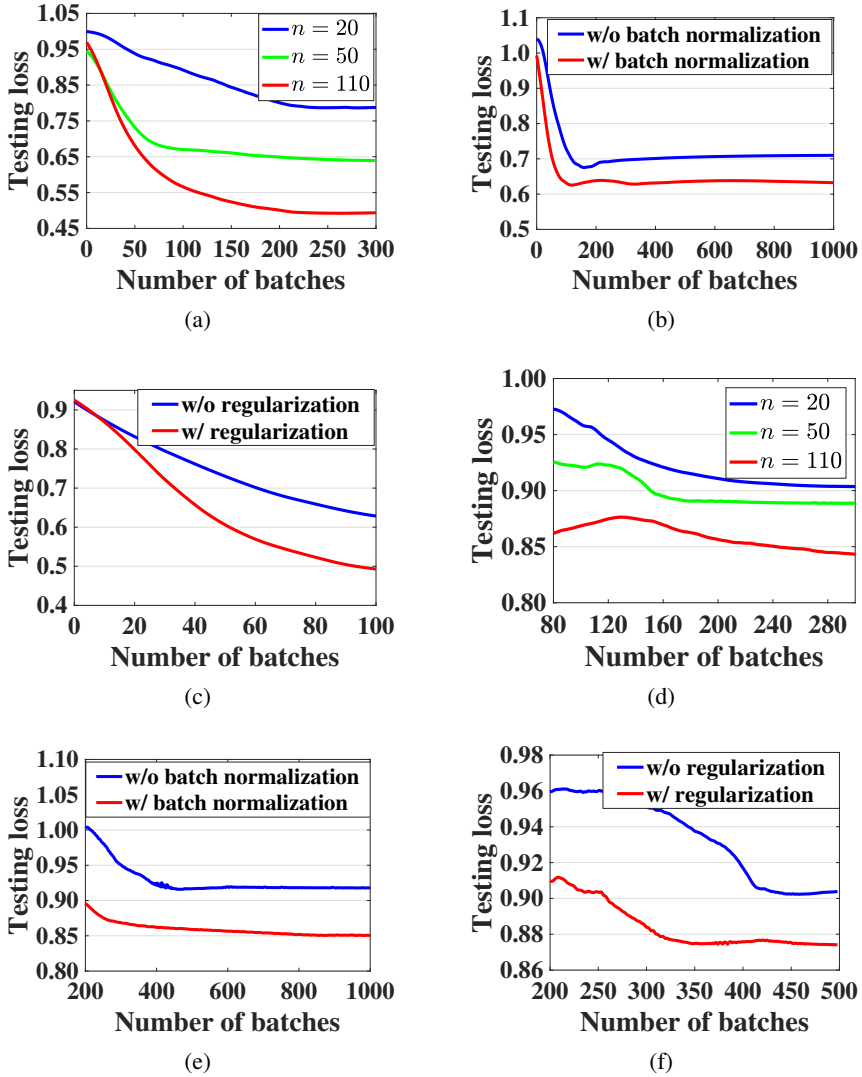
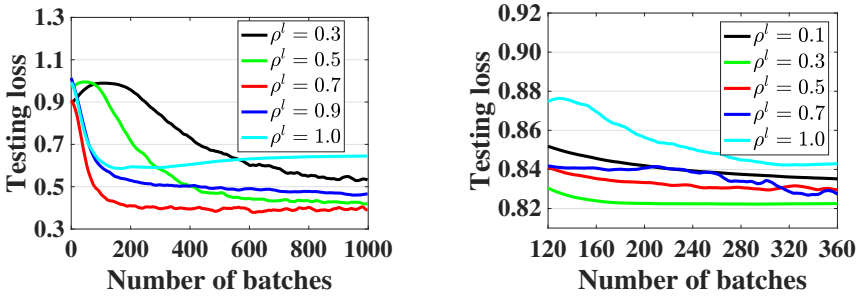


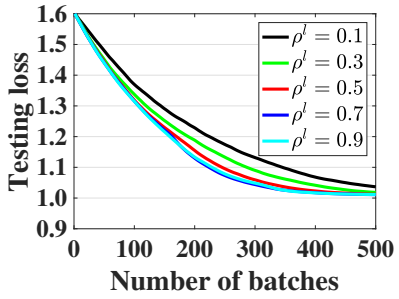
Figure 3.1: The testing loss of the DML model on the MNIST 8v9 dataset (a-c) and the bone disease dataset (d-f). (a) and (d): The effect of the training data size. (b) and (e): The effect of batch normalization. (c) and (f): The effect of regularization.

Then we report the calculated testing losses on the MNIST 8v9 and bone disease datasets in Figure 3.1c and Figure 3.1f, respectively. From the two figures we can see the models with regularization perform much better than those without regularization. That is to say, regularization can help to improve the generalization performance.

The effect of dropout. Next, we analyze the effect of dropout on the performance of the DML model. In this experiment, we adopt the MNIST 8v9, bone disease and wine quality datasets. We vary the retention rate from 0.3 to 1.0 for the MNIST 8v9 dataset, from 0.1 to 1.0 for the bone disease dataset and from 0.1 to 0.9 for the wine quality dataset. Note that when the retention rate for each layer is set as 1.0 (i.e., $\forall l \in [L], \rho^l = 1.0$), it is the case without dropout. The calculated testing losses for the three datasets are shown in Figure 3.2. The results show that the DML model trained with dropout performs better than that trained without dropout, which means dropout can improve the model's generalization ability. Additionally, from this figure we can see that smaller retention rate does not mean better generalization performance. This result also accords with the previous theoretical analysis.



(a) MNIST 8v9 (b) Bone disease



(c) Wine quality

Figure 3.2: The testing loss of the DML model under different retention rates on the MNIST 8v9, bone disease and wine-quality datasets.

The effect of the input feature dimension. Finally, we evaluate the effect of the input feature dimension (i.e., d) on the training loss. Here we adopt the MNIST 8v9 dataset. We first randomly select a subset of features to reduce the feature dimension of this dataset. Then, for

this newly derived reduced-dimensional dataset, we randomly select $1,200$ samples as the training samples (i.e., $n = 1,200$). Additionally, we consider three neural network structures in this experiment, and all of them have three layers. The numbers of the units in different layers of the three network structures are (d, 64, 10), (d, 80, 10) and (d, 128, 16), respectively. Figure 3.3 reports the evolution of the training loss under various input feature dimensions (i.e., d). In this figure, each line denotes the evolution of the training loss for a specific value of d. We can see that the smaller the value of of the input feature dimension (i.e., d), the larger the training loss. This is also consistent with our previous theoretical analysis in Section 3 that when the value of the input feature dimension (i.e., d) is very small, the empirical training loss could be very large. The reason is that when the number of the randomly selected features is very small compared with that in the original unreduced dataset, most of the useful information in the original dataset cannot be captured, which means the learned DML model cannot capture the particular characteristic features in the original dataset. Thus, the empirical training loss becomes large.

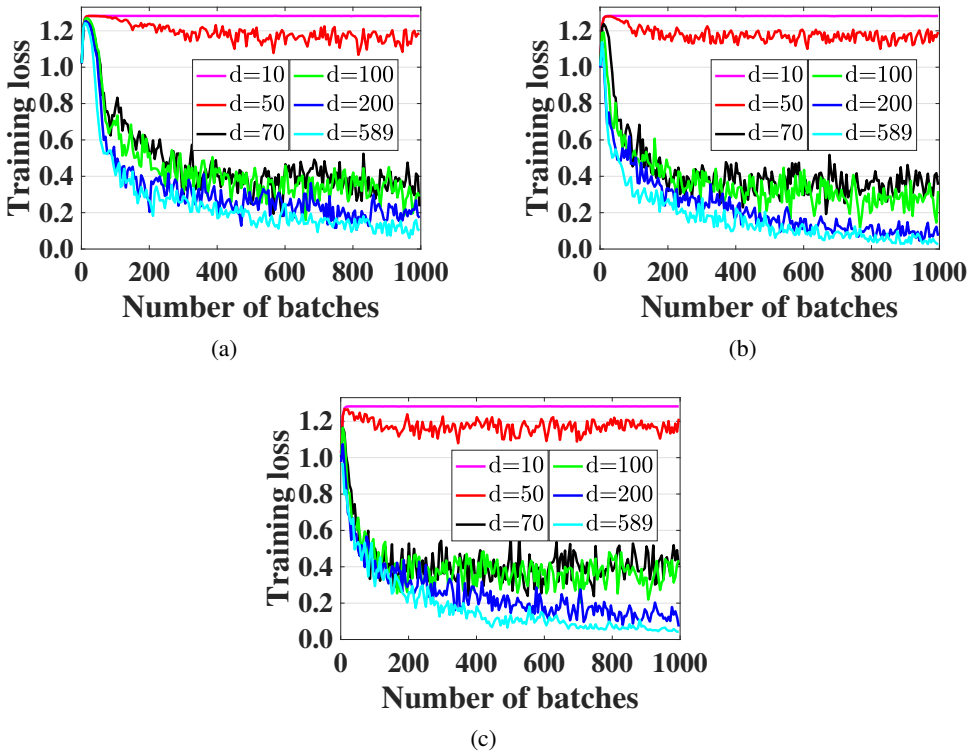


Figure 3.3: The training loss of the DML model for different input feature dimensions on the MNIST 8v9 dataset. The results in (a), (b) and (c) are for three different neural network structures, respectively.

3.5.3 Experiments for ADroDML

In this section, we evaluate the performance of ADroDML on the MNIST 8v9 and wine quality datasets. For each dataset, we first select 1,200 samples as the training set, and then use the remaining samples as the test set.

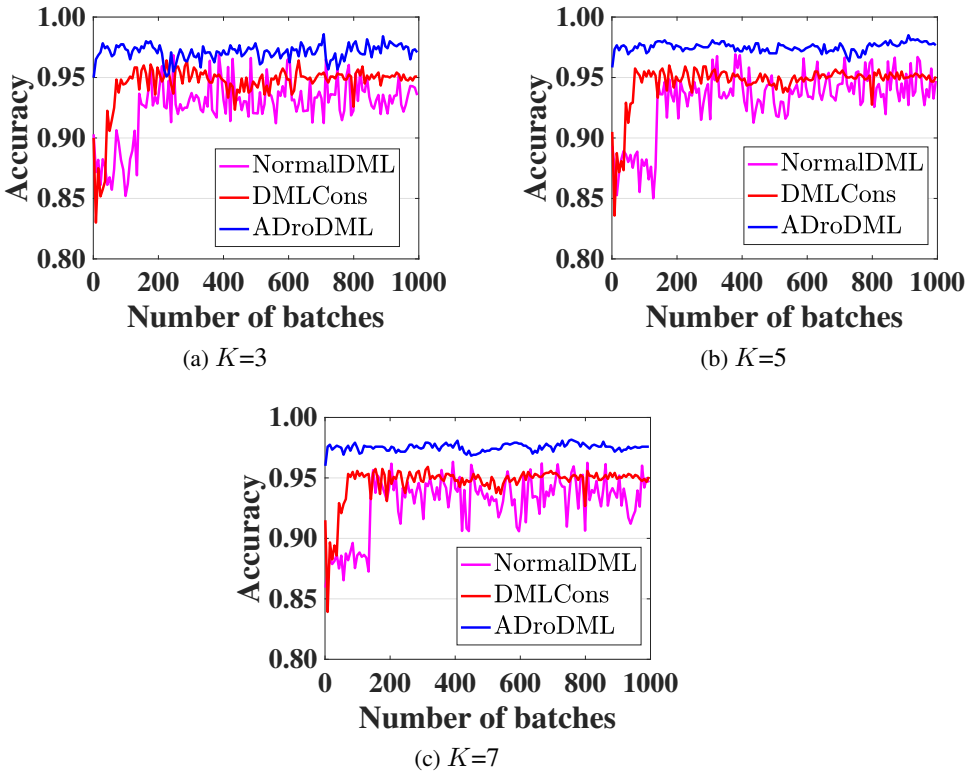
Baseline methods. We compare the performance of ADroDML with the following two baseline methods:

- *Normal DML training (NormalDML).* In this method, we use the standard contrastive loss to train a DML model and do not take the dropout strategy into consideration.
- *DML training with a constant dropout retention rate (DMLCons).* In this method, we consider the dropout strategy with a pre-defined retention rate. Here we set the value of the retention rate as 0.5 by following existing DML works [53, 54].

For the sake of fairness, the network structure for each of the baseline methods is the same as that of ADropDML.

Performance. In this experiment, we use the standard K-nearest neighbor algorithm (KNN) as the classifier, which means for each given test sample, its label is assigned by majority voting over its top-K nearest samples in the training set. Here we consider three cases where the value of K is set as 3, 5 and 7, respectively. In Figure 3.4 and Figure 3.5, we respectively report the classification accuracy of ADroDML on the MNIST 8v9 dataset and the wine quality dataset. The results in the two figures show that our proposed ADroDML can achieve the best performance in all cases. When $K = 3$, the classification accuracy of ADroDML on the wine quality dataset is around 97.8% while that of the two baseline methods (i.e., NormalDML and DMLCons) is around 60.0% and 62.8%, respectively. The main reason is that the proposed ADroDML can adaptively learn the optimal dropout retention rates to avoid the overfitting problem.

Convergence. Next, we evaluate the convergence of ADroDML through calculating the training loss in each batch of the training process. Figure 3.6 reports the experimental results on the MNIST 8v9 dataset. Here we conduct the experiment for three times. Each time the training data are randomly selected from the dataset. From this figure, we can see that the training loss gradually converges to zero with the increase of the number of batches. This confirms that the convergence can be guaranteed in our proposed method ADroDML.



(a) K=3 (b) K=5

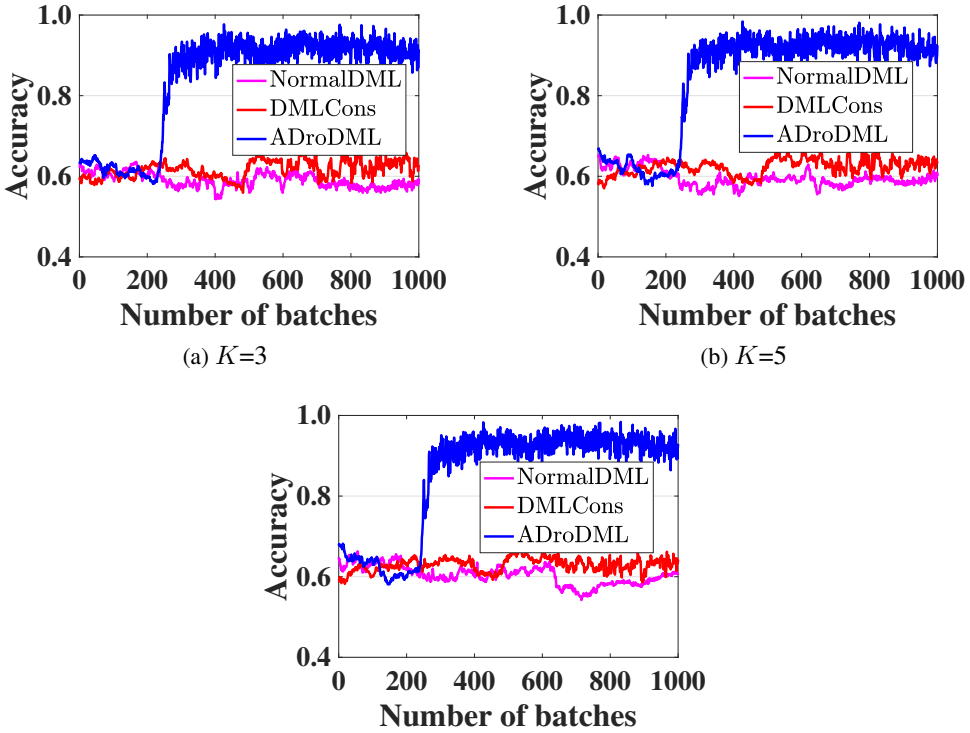
(c) K=7

Figure 3.4: Classification accuracy of the proposed ADroDML on the MNIST 8v9 dataset.

3.6 Related Work

Based on the types of neural networks, the existing DML works can be roughly divided into the following three categories: (1) **The Siamese network based DML methods** [37, 38] are trained by minimizing a contractive loss function, where the task is to minimize the distance between similar sample pairs and to push the pairwise distance between dissimilar pairs larger than a fixed margin. (2) **The triplet network based DML methods** [39, 40] are trained by minimizing a triplet loss function, and the triplets are usually generated based on the class labels of the training dataset. (3) There are also some **DML methods based on other types of networks** [41, 42]. However, all the existing works do not provide generalization analysis for DML. Additionally, they do not consider how to derive the optimal retention rates for dropout.

Although there are some works providing the theoretical analysis for traditional linear metric learning [55, 56], they can not be directly used for analyzing the generalization bound of



(a) K=3 (b) K=5

(c) K=7

Figure 3.5: Classification accuracy of the proposed ADroDML on the wine quality dataset.

DML. Even though [56] also adopts the Rademacher complexity, this work is significantly different from ours. First of all, we provide the generalization bound for DML that solves the *nonlinear transformation* problem, while [56] presents the bound for traditional metric learning that solves the *linear transformation* problem. Thus, the problem studied here is more challenging than that in [56]. Secondly, to make the theoretical analysis more general, we take into account *the dropout strategy* for DML models, which is unfortunately not considered in [56].

Additionally, adding a regularization related to a specific upper bound to learn the parameters in an optimal way has many practical applications [57, 46]. However, their settings are different from ours. For example, [46] considers the classification model with a final softmax layer, which assumes that the input data are i.i.d. However, we aim to derive the bound for DML, where the input data are not independent, making our problem more challenging. Moreover, *the techniques used to derive the bound* here and those used in [46] are significantly different. In particular, the derived bound based on the techniques in [46] has an exponential dependency

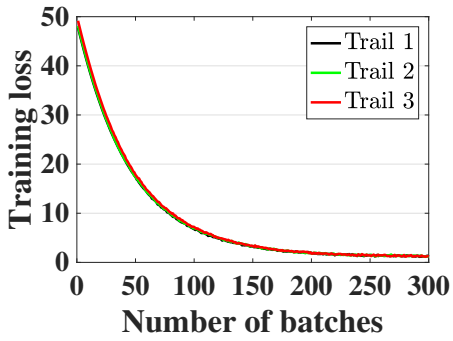


Figure 3.6: The training loss of AdroDML w.r.t Number of batches on the MNIST 8V9 dataset.

(i.e., 2^L) on the network length L, which is not appealing even for moderate networks. In contrast, based on our proposed techniques, the derived bound only linearly depends on L. Finally, *the dropout strategy* considered here and that in [46] are different. Here, we consider randomly dropping out each connection, whereas they considered randomly dropping out each hidden neuron, which is more restrictive than ours.

3.7 Conclusions

In this chapter, we present the generalization error bound for DML and analyze the findings derived from this bound. Additionally, we propose a novel method (ADroDML) to adaptively adjust the dropout rates for DML based on the derived generalization bound. Compared with existing DML works that require predefined dropout rates, ADroDML can adaptively learn the dropout retention rates for DML in a theoretically justified way. We also conduct experiments on real-world datasets to verify the findings derived from the generalization bound and test the effectiveness of the proposed adaptive method. The experimental results are consistent with our theoretical analysis, and they also show that the proposed ADroDML can achieve much better performance compared with the baselines.

Part II

Studying Security Vulnerability and Robustness to Malicious Attacks

Chapter 4

On the Robustness of Deep Reinforcement Learning Interpretations

4.1 Introduction

In recent years, there has been increasing interest in RL, a machine learning paradigm that has achieved great success in addressing challenging sequential decision-making problems [58]. The key components of RL include an agent and its environment, where the agent learns an optimal action selection policy by iteratively interacting with and receiving rewards from its environment. RL has served in a wide spectrum of applications, such as healthcare, autonomous navigation, and optimal control. In reality, many achievements of RL are due to its combination with deep learning. This combination, called DRL, is more capable of handling tasks with either high dimensional state space or complex task selection policy. Recently, various DRL algorithms have been developed, including deep Q-networks (DQN) [59], trust region policy optimization (TRPO) [60], and asynchronous advantage actor-critic (A3C) [61].

Although DRL techniques have shown superior performance in many real-world applications, their decision-making process is opaque and lacking transparency, which makes the inner workings of DRL models incomprehensible for human users. The "black box" nature of the DRL models may impede users from trusting the predicted results, especially when the model is used for making critical decisions (e.g., medical diagnosis and autonomous driving), because the consequences may be catastrophic if the predictions are acted upon blind faith. To address

this problem, a plethora of DRL interpretation methods have been proposed to gain insight into the decision-making process of DRL agents [59, 62, 63, 61, 64, 65, 66]. These interpretation methods can provide explanations for particular predictions of DRL models and help humans understand the inner mechanisms. For example, at each time step of a sepsis patient's trajectory, doctors can use these interpretation methods to interpret the clinical decision making. However, an implicit assumption for these DRL interpretation methods is that they are performed in a reliable and secure environment, which may not be true in practice. As DRL interpretations play an increasingly critical role in many real-world applications, they are susceptible to a risk of being maliciously attacked. In this chapter, we consider two representative types of attacks against DRL interpretations: *adversarial attacks* and *model poisoning attacks*.

Adversarial attacks. This type of attacks happens in the testing stage of DRL where the attacker tampers the input data after the victim DRL model is trained. Unlike deep supervised learning models, where decision is made instantaneously and independently, adversarial attacks to DRL models are extremely difficult to analyze quantitatively and defend effectively, as DRL models involve a temporally dependent sequential decision making process where states at different time-steps are perturbable. At each targeted victim time-step, the attacker's goal is to fool both the DRL model and the corresponding DRL interpretation method through manipulating the current state observation that is communicated between the agent and the environment. For example, at a specific time step of a sepsis patient's trajectory, the attacker could add adversarial perturbation onto patient's clinical records, which not only causes the DRL model to produce wrong medical decision but also leads the adopted interpreter to give wrong interpretation results.

Model poisoning attacks. Different from the adversarial attacks at the testing stage, the model poisoning attacks occur in the training stage of DRL. In this type of attacks, the attacker aims to dramatically degrade the performance of the DRL interpretation methods through manipulating the learned DRL model parameters, while maintaining the original performance of DRL model to ensure maximal stealthiness. This type of attacks is common in many real-world applications. Consider the example where due to the vast computational cost of training DRL models, the agent resorts to downloading the well-trained DRL model from an online model repository to complete its own tasks. However, during this process, the attacker could perform the model poisoning attack to manipulate the pre-trained DRL model to make the agent unwittingly download a maliciously re-trained DRL model.

Despite the prevalence of malicious attacks in real-world applications, there is no existing work studying the vulnerability of DRL interpretations to these attacks. Although there are some

works addressing the adversarial vulnerability of the interpretation methods for supervised deep neural networks (DNN) based classification models [67, 68, 69, 70, 71], they cannot be directly applied to DRL interpretations due to the following unique features of DRL: First of all, they only focus on attacking a particular test instance in supervised classification settings, and ignore the agent's sequential decision-making process that consists of a continuous sequence of state-action predictions. If we directly adopt these per-instance attack methods and craft different perturbations to different states, the computation complexity will be extremely high. Secondly, a human imperceptible per-instance attack as in supervised deep learning models, may be highly noticeable for DRL agent as a tiny perturbation in one certain state may have unpredictable and apparent shift towards the whole future path. DRL is in fact goal-oriented, and it aims to learn sequences of actions that can lead the agent to achieve its goal. For example, in the autonomous driving scenarios, the ultimate goal of the agent is to successfully and safely reach the desired destination. If an attacker locally perturbs some states with per-instance perturbation methods without a global grasp of agent's end goal, the agent can easily detect the attack based on the deviation away from his desirable destination. Thus, the attacker who aims to attack the DRL models should be more cautious because he needs not only to guarantee the imperceptibility of local perturbations, but also to avoid compromising the end goal of the agent. Last but not least, an implicit assumption in the existing adversarial attack methods is that the attacker has the capability of manipulating the whole input data (e.g., the entire image). However, in practice, the attacker may be restricted to only manipulating a subset of each input data point (e.g., the bottom right region where digital watermark is), and hence this assumption may be impractical for the real-world physical attacks.

As for the model poisoning attacks, to the best of our knowledge, there is no existing work studying the vulnerability of interpretation models to such attacks. The challenge here is how to manipulate the pre-trained DRL model such that the attacker can dramatically alter the interpretation results without significantly hurting the performance of the DRL model. If the performance of the DRL model is significantly degraded after the manipulation, the model poisoning attack can be easily detected by evaluating on a holdout set, and then the manipulated DRL model will be immediately rejected by the agent.

To well understand the performance of DRL interpretations in malicious environment, in this chapter, we study their vulnerability to the above two types of attacks. Specifically, we first propose an universal adversarial attack against DRL interpretations (**UADRLI**), based on which the attacker can add the crafted universal perturbation to the environment states on a maximum number of time steps while incurring minimal damage to the agent's end goal. In our design,

the optimal attack strategy can be efficiently derived by solving an optimization problem even with the sequential and progressive nature of DRL taken into account. Additionally, we propose a model poisoning attack against DRL interpretations (**MPDRLI**), based on which the attacker can manipulate the pre-trained DRL model such that the attacker can dramatically alter the interpretation results without significantly hurting the efficacy of the original DRL model. With the proposed model poisoning manipulation, the interpretations can be successfully misled in multiple experimental settings. We also provide theoretical results indicating the change in overall efficacy of DRL model is strictly bounded, which guarantees the difficulty to detect such attacks. In order to mitigate the effect of malicious attacks, we also propose a general defense mechanism to increase the attack resistance of DRL interpretations against the malicious attacks. To summarize, our contributions are:

- First of all, we propose an universal adversarial attack against DRL interpretations (i.e., UADRLI), which aims to craft a single universal perturbation that can be applied identically (uniformly) on every time step. Based on the proposed UADRLI, the attacker can efficiently deceive downstream DRL interpretation methods via state perturbations.
- We also design a model poisoning attack against DRL interpretations (MPDRLI), based on which the attacker can secretly alter the interpretation results through providing the agent a strategically poisoned but equally effective pre-trained DRL model.
- In order to enhance the robustness of DRL interpretations against malicious attacks, we also propose a general defense mechanism to increase the attack resistance of DRL interpretations against the malicious attacks.
- Both theoretical analysis and extensive experimental results validate the effectiveness of the proposed approaches.

4.2 Preliminary

Deep reinforcement learning. In RL, an agent aims to learn an optimal behavior through trial-and-error by sequentially interacting with an environment, which is referred to as the Markov Decision Process (MDP) defined with a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$. In a MDP, the agent interacts with its environment in the following way: The agent starts by gathering an initial state $\boldsymbol{s}_0 \in \mathcal{S}$ that describes the environment. At each time step t, the agent chooses an action $a_t \in \mathcal{A}$ according to some policy π based on the current state $\boldsymbol{s}_t \in \mathcal{S}$ (i.e., $a_t = \arg\max_{a \in \mathcal{A}} \pi(\boldsymbol{s}_t, a)$),

and progresses to a new state $\boldsymbol{s}_{t+1} \in \mathcal{S}$ according to the transition dynamics $\mathcal{P}$. Additionally, the agent receives a scalar reward $r(\boldsymbol{s}_t, a_t) = \mathcal{R}(\boldsymbol{s}_t, a_t)$, provided by the reward function $\mathcal{R}$, which judges the quality of its decision. This sequential decision making process produces a sequence of state-action pairs $\mathbb{T} = \{(s_t, a_t)\}_{t=0}^{T}$, where T is the timestep that the environment terminates. The return is computed as $R = \sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t, a_t)$, where $\gamma \in [0, 1]$ is a discount factor indicating how much the agent values an immediate reward compared to a future reward. The agent's goal is to find a policy π^* that maximizes the expected value of the total reward from all states

$$\pi^* = \arg\max_{\pi} \{\mathbb{E}[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t, a_t)]\}, \tag{4.1}$$

where $\mathbb{E}$ denotes the expectation over all possible trajectories generated by policy π. DRL is the combination of deep learning and RL, and is proposed to overcome the challenges in learning control policies from high-dimensional raw input data and large state and action spaces in traditional RL environments. In DRL, we represent the policy π with a deep neural network that is parameterized by $\boldsymbol{\Theta}$ (i.e., the weights of the policy network). To find out the optimal policy parameters, many different DRL algorithms have been proposed, including DQN [59], TRPO [60], and A3C [61].

Interpretation methods for deep reinforcement learning. Currently, many works have been proposed to make DRL more transparent. In general, existing interpretation methods for DRL can be grouped into two categories: intrinsic interpretability and post-hoc interpretability. The latter case does not require modifying model architectures or parameters, thereby leading to higher prediction accuracy. In this chapter, we mainly consider post-hoc interpretations. Formally, for the given state-action pair $(\boldsymbol{s}_t, a_t)$ and policy π, the post-hoc interpreter $\mathcal{I}$ can generate the feature importance scores $\mathcal{I}(\boldsymbol{s}_t, a_t; \pi)$, which measures how important each feature of the state $\boldsymbol{s}_t$ is, in determining the corresponding action a_t under policy π. In the following, we describe several widely-used post-hoc interpretation methods for DRL, all of which aim to generate feature importance scores (also called saliency maps) to show the relevancy of each feature for the prediction.

- **Gradient saliency.** This method [61] is a generic interpretation method that combines gradient information with class activation maps to visualize the importance of each feature. The map is computed as $\mathcal{I}(\boldsymbol{s}_t, a_t; \pi) = \frac{\partial \pi(\boldsymbol{s}_t, a_t)}{\partial \boldsymbol{s}_t}$, and quantifies how sensitive the action prediction score (i.e., $\pi(\boldsymbol{s}_t, a_t)$) is with respect to the small changes of input features.

- **Jacobian saliency.** Wang et al. [62] extend gradient-based saliency maps to DRL by computing the Jacobian of the output logits with respect to a stack of input images. Specifically, to visualize the salient part of the image as seen by the value stream, they compute the absolute value of the Jacobian of the predicted state value with respect to the input frame.

- **Object saliency.** Iyer et al. [72] use template matching, a common computer vision technique, to detect objects within an input image and measure saliency through changes in Q-values for masked and unmasked objects.

- **Perturbation saliency.** Greydanus et al. [61] use saliency maps to provide explanations for the DRL agent's behaviors over temporally extended sequences. Specifically, they generate saliency maps by perturbing the original input image using a Gaussian blur of the image and measure changes in policy from removing information from a region.

Note that all of the above interpretation methods for DRL focus on instance-level interpretability, which means the different interpretation results would be given on different state-action pairs in one episode, and provide the importance score of each feature.

4.3 Adversarial Attack against DRL Interpretations

In this section, we first introduce the threat model and then develop an optimization framework to formalize our universal adversarial attack against DRL interpretations (i.e., UADRLI). After that, we present the theoretical analysis for the proposed universal attack.

4.3.1 Threat Model

Following the line of work on adversarial attacks [67, 68, 69, 70, 71], we here assume a white-box setting, which is a conservative and realistic assumption. The attacker in this setting tries to evade the system by manipulating malicious states during the testing phase. The attacker cannot change the DRL algorithm used for the training of the agent, and cannot change the architecture of the policy networks. The attacker can only change the state observations that are communicated between the agent and the environment. The attacker's goal is to deceive both the trained DRL model and its adopted interpretation method.

4.3.2 Formalization of Universal Adversarial Attack

In the adversarial attack settings, when we design the adversarial attack against DRL interpretations, we need to take the unique characteristics of DRL into account. Firstly, the attacker should handle the sequentiality of DRL, and this sequential decision-making process produces a sequence of state-action pairs $\{(\boldsymbol{s}_t, a_t)\}_{t=0}^{T}$. If we directly adopt existing methods on supervised learning attacks and craft state-dependent adversarial perturbations, the computation complexity will be significantly increased due to the generation of a large amount of perturbations different with each other. Additionally, when generating state-dependent perturbations, the attacker has to query the victim DRL model at test time. Instead, we propose to add the state-agnostic perturbations. Specifically, we aim to craft a single universal adversarial perturbation (dubbed as $\boldsymbol{\delta}$), which can be identically (uniformly) applied on every observed state without accessing the target victim DRL model at test time. We further restrict the attacker to only manipulating pixels within a small region of the image-the attacker may choose the location of the area, but cannot perturb pixels outside this selected image area. Formally, for any given observed state $\boldsymbol{s}_t$, the attacker crafts the corresponding adversarial state $\tilde{\boldsymbol{s}}_t$ as follows

$$\tilde{\boldsymbol{s}}_t = A(\boldsymbol{s}_t, k_t * \boldsymbol{M}, \boldsymbol{\delta}) = (1 - k_t * \boldsymbol{M}) \odot \boldsymbol{s}_t + (k_t * \boldsymbol{M}) \odot \boldsymbol{\delta}, \tag{4.2}$$

where $k_t \in \{0, 1\}$, $\odot$ denotes the matrix element-wise product, and $\boldsymbol{\delta}$ is the universal perturbation to be generated. Here, $\boldsymbol{M}$ is a predefined binary mask matrix representing the position and shape of the area that can be attacked. Note that $k_t \in \{0, 1\}$ denotes whether at time step t the perturbation should be applied ($k_t = 1$) or not ($k_t = 0$). Specifically, if $k_t = 1$, $A(\boldsymbol{s}_t, k_t\boldsymbol{M}, \boldsymbol{\delta}) = (1 - \boldsymbol{M}) \odot \boldsymbol{s}_t + \boldsymbol{M} \odot \boldsymbol{\delta}$. Otherwise, $A(\boldsymbol{s}_t, k_t\boldsymbol{M}, \boldsymbol{\delta}) = (1 - (0 * \boldsymbol{M})) \odot \boldsymbol{s}_t + (0 * \boldsymbol{M}) \odot \boldsymbol{\delta} = 1 \odot \boldsymbol{s}_t = \boldsymbol{s}_t$. Secondly, we should make sure that the universal adversarial perturbation (i.e., $\boldsymbol{\delta}$) that causes the wrong predictions is imperceptible. We use parameter ϵ to control the magnitude of the universal perturbation $\boldsymbol{\delta}$. Specifically, given a state $\boldsymbol{s}_t$ of the system, the attacker can only select a perturbed state $\tilde{\boldsymbol{s}}_t$ as

$$\tilde{\boldsymbol{s}}_t \in \{\tilde{\boldsymbol{s}}_t \in \mathcal{S} : d(\boldsymbol{s}_t, \tilde{\boldsymbol{s}}_t) \leq \epsilon\}, \tag{4.3}$$

where $d(\boldsymbol{s}_t, \tilde{\boldsymbol{s}}_t) = \|\boldsymbol{s}_t - \tilde{\boldsymbol{s}}_t\|_\infty = \|\boldsymbol{\delta}\|_\infty$. Thirdly, when added to any clean state $\boldsymbol{s}_t$, the universal adversarial perturbation will cause the policy to select a different action at this state and the interpretation results to be wrong at the same time. Note that the attacked image region is the real reason why the policy alters decision. Hence, the attacker should mislead the interpreter to highlight other image regions that are not perturbed. In addition to fooling the interpreter for

DRL, when adding $\boldsymbol{\delta}$ to state $\boldsymbol{s}_t$ at time step t, the attacker also wants to mislead the agent to take any wrong action (instead the original optimal action), that is,

$$\arg\max_{a\in\mathcal{A}} \pi(\boldsymbol{s}_t, a) \neq \arg\max_{a\in\mathcal{A}} \pi(\tilde{\boldsymbol{s}}_t, a), \tag{4.4}$$

where $\tilde{\boldsymbol{s}}_t$ is calculated based on Eqn. (4.2). Last but not least, since DRL is goal-oriented, the attack will be easily detected if the attacker perturbs the observed state at every time step. The reason is that when the attacker perturbs every observed state, the final accumulated reward will be largely compromised. On the attacker's side, he also wants to maximize his expected utility. In this chapter, we consider the case where the attacker wants to maximize the total number of attacked time steps (i.e., $\sum_{t=0}^{T} k_t$). On the other hand, to avoid being detected, the attacker should also make sure that the end goal of the entire DRL task is not significantly compromised. In practice, the end goal of the agent is formalized in terms of the accumulated reward in the long run. Specifically, the attacker should make sure that the difference of the accumulated reward before and after the attack should be small, and the difference is defined as $(\sum_{t=0}^{T} \mathbb{E}_\pi[\gamma^t r(\tilde{\boldsymbol{s}}_t, \tilde{a}_t)] - \sum_{t=0}^{T} \mathbb{E}_\pi[\gamma^t r(\boldsymbol{s}_t, a_t)])^2$, where $\tilde{a}_t = \arg\max_{a\in\mathcal{A}} \pi(\tilde{\boldsymbol{s}}_t, a)$. Now, the problem is how can the attacker find an effective attack strategy $\{k_0, \cdots, k_T\}$ with the corresponding single universal perturbation $\boldsymbol{\delta}$, which not only maximizes the attacker's utility (i.e., $\sum_{t=0}^{T} k_t$) but also incurs minimal damage to the agent's end goal.

Based on the above arguments, the attacker's goal is to add *smallest (imperceptible)* universal perturbation to the environment states in a *maximum* number of steps while incurring *minimal* damage to the agent's end goal. With this goal in mind, for the given episode that consists a sequence of state-action pairs $\{(\boldsymbol{s}_t, a_t)\}_{t=0}^{T}$ and the threat model, at high level, we formulate the proposed adversarial attack using the following optimization framework

$$\begin{aligned}
\min_{\boldsymbol{\delta}, \{k_t\in\{0,1\}\}_{t=0}^{T}} & \Big(\sum_{t=0}^{T} \mathbb{E}_\pi[\gamma^t r(\tilde{\boldsymbol{s}}_t, \tilde{a}_t)] - \sum_{t=0}^{T} \mathbb{E}_\pi[\gamma^t r(s_t, a_t)]\Big)^2 \\
& + \lambda_1 \sum_{t=0}^{T} \exp(\mathcal{I}(\tilde{\boldsymbol{s}}_t, \tilde{a}_t; \pi) \odot \boldsymbol{M}) - \lambda_2 \sum_{t=0}^{T} k_t \\
s.t. \quad & \forall t \in [T], \tilde{\boldsymbol{s}}_t = A(\boldsymbol{s}_t, k_t * \boldsymbol{M}, \boldsymbol{\delta}) \\
& \forall t \in [T], \tilde{a}_t = \arg\max_{a\in\mathcal{A}} \pi(\tilde{\boldsymbol{s}}_t, a), \\
& \|\boldsymbol{\delta}\|_\infty \leq \epsilon, \\
& \forall t \in \mathcal{T}_1, \arg\max_{a\in\mathcal{A}} \pi(\boldsymbol{s}_t, a) \neq \arg\max_{a\in\mathcal{A}} \pi(\tilde{\boldsymbol{s}}_t, a),
\end{aligned} \tag{4.5}$$

where $\mathcal{T}_1 = \{t : k_t = 1\}$ denotes the set of time steps being attacked, and λ_1 and λ_2 are two regularization parameters. The first loss term is utilized to enforce that the actual ultimate accumulated reward does not change significantly. The second loss term is used to decrease the importance scores of the pixels that are within the attacked image region. The third loss term is introduced to maximize the attacker's utility (i.e., the number of attacked time steps). The two hyper-parameters (i.e., λ_1 and λ_2) balance the three factors. The third constraint allows the attacker to manipulate all the states that the agent perceives within an ϵ budget, and hence ensures that the universal perturbation (i.e., $\boldsymbol{\delta}$) is imperceptible. The last constraint forces that the action predictions are wrong.

4.3.3 Optimization

In this section, we discuss how to solve the optimization problem described in Eqn. (6). However, it is very challenging to directly solve the above optimization problem as it involves too many variables. To address this challenge, we propose to convert it into two sub-problems, and then solve them in two separate steps. Specifically, in the first step, we aim to figure out the universal adversarial perturbation $\boldsymbol{\delta}$ by solving a sub-optimization problem. In the second step, we solve the problem of identifying the optimal attack strategy $\{k_0, \cdots, k_T\}$. Below, we elaborate the two steps in greater detail.

Step 1: Generating the universal adversarial perturbation. In this step, we focus on how to generate the universal adversarial perturbation (i.e., $\boldsymbol{\delta}$), which can be applied identically on every time step. As aforementioned, when generating the universal adversarial perturbation $\boldsymbol{\delta}$, the attacker should satisfy the following requirements: Firstly, when applying the universal adversarial perturbation (i.e., $\boldsymbol{\delta}$) to each observed state, the attacker should make sure that not only the predicted action is altered but also the corresponding interpretations are wrong. Additionally, we should note that the attacker is restricted to only manipulating pixels within a small region of the input image. Based on the above two restrictions, for the given unattacked trajectory $\{(\boldsymbol{s}_t, a_t)\}_{t=0}^{T}$, we formulate the following optimization problem

$$\begin{aligned}
\min_{\boldsymbol{\delta}} \sum_{t=0}^{T} \frac{1}{T+1} \pi(\tilde{\boldsymbol{s}}_t, a_t) + \lambda_1 \sum_{t=0}^{T} \frac{1}{T+1} \exp(\mathcal{I}(\tilde{\boldsymbol{s}}_t, \tilde{a}_t; \pi) \odot \mathcal{M}) \\
s.t. \quad \forall t \in [T], \tilde{\boldsymbol{s}}_t = A(\boldsymbol{s}_t, \boldsymbol{M}, \boldsymbol{\delta}), \\
\forall t \in [T], \tilde{a}_t = \arg\max_{a \in \mathcal{A}} \pi(\tilde{\boldsymbol{s}}_t, a), \\
\|\boldsymbol{\delta}\|_\infty \leq \epsilon,
\end{aligned} \tag{4.6}$$

where $A(\boldsymbol{s}_t, \boldsymbol{M}, \boldsymbol{\delta}) = (1 - \boldsymbol{M}) \odot \boldsymbol{s}_t + \boldsymbol{M} \odot \boldsymbol{\delta}$. The first loss term is used to force the agent to take an arbitrary action (instead of the original optimal action). The second term is used to fool the interpretation results. The first constraint enforces the attacker to only manipulate a small region of the input data. The last constraint aims to find the sufficiently imperceptible universal perturbation that leads to the wrong action prediction and interpretations desired by the adversary. To derive the universal perturbation $\boldsymbol{\delta}$, we can solve the above optimization problem using the projected gradient algorithm.

Step 2: Identifying attack points. Note that after Step 1, the attacker can generate the universal perturbation $\boldsymbol{\delta}$ that can be identically applied to every time step. However, as aforementioned, if the attacker perturbs the observed state at every time step, the launched universal attack will be easily detected due to the significant decrease in the end reward. Hence, in this step, we discuss how to identify the optimal attack strategy $\{k_0, \cdots, k_T\}$ by strategically selecting a set of time steps. With this identified attack strategy, the attacker can maximize his attack utility while avoiding being detected. To derive the attack strategy, at each time step, the attacker first computes the variance of the Q value as follows

$$Var(Q(\boldsymbol{s}_t)) = \frac{1}{|\mathcal{A}| - 1} \sum_{i=1}^{|\mathcal{A}|} (Q(\boldsymbol{s}_t, a_i) - \frac{1}{|\mathcal{A}|} \sum_{j=1}^{|\mathcal{A}|} Q(\boldsymbol{s}_t, a_j))^2, \tag{4.7}$$

where $\mathcal{A}$ is the action space of the MDP, and $|\mathcal{A}|$ is the number of actions. Then, according to the above calculated variance, the attacker decides whether he should perturb $\boldsymbol{s}_t$. Based on Lemma 5 (in Appendix), we know that when attacking states with low variance, the attacker will get more reward in expectation. Hence, to avoid being detected, the attacker should attack the states with low variance to incur low decrease in the accumulated reward.

4.3.4 Theoretical Analysis

In this section, we theoretically quantify the influence of our proposed universal attack on the accumulated reward collected by the agent throughout the game. To do so, we first characterize the environment under attack as a new MDP[1] (denoted as $\mathcal{M}_1$ or $\mathcal{M}_2$, depending on detailed attack setup), which is different from the original MDP $\mathcal{M}$ in its transition probability and immediate reward function. Then we have the following theorem.

[1] Due to space constraints, both the formal definitions of $\mathcal{M}_1$ and $\mathcal{M}_2$, and the proof of the theorems are deferred to Appendix.

Theorem 4. *Let V^*, V_1^*, and V_2^* be optimal value functions for $\mathcal{M}$, $\mathcal{M}_1$, and $\mathcal{M}_2$, respectively. Note that the value function is equal to the expected total reward for an agent starting from a particular state. Suppose $\mathcal{M}$, $\mathcal{M}_1$, and $\mathcal{M}_2$ have bounded immediate rewards, i.e., $\max_{\boldsymbol{s}\in\mathcal{S},a\in\mathcal{A}} |\mathcal{R}(\boldsymbol{s},a)| \leq R$. Let $TV(\mathcal{P},\mathcal{Q})$ be the total variation distance between two probability measures $\mathcal{P}$ and $\mathcal{Q}$ on $\mathcal{S}$. Let $\|f-g\|_\infty = \max_{\boldsymbol{s}\in\mathcal{S},a\in\mathcal{A}} |f(\boldsymbol{s},a)-g(\boldsymbol{s},a)|$. Suppose $\mathcal{M}$ has transition and immediate reward models which are continuous on $\mathcal{S}$, i.e., $\forall \boldsymbol{s}\in\mathcal{S}, a\in\mathcal{A}$, $\|\mathcal{P}(\cdot|\boldsymbol{s},a) - \mathcal{P}_1(\cdot|\boldsymbol{s},a))\|_1 \leq L\epsilon$, and $|\mathcal{R}(\boldsymbol{s},a) - \mathcal{R}_2(\boldsymbol{s},a)| \leq l\epsilon$, for some constant L and l. Then, we have*

$$
\begin{aligned}
\|V^* - V_1^*\|_\infty &\leq \frac{2\gamma R}{(1-\gamma)^2} \max_{\boldsymbol{s}\in\mathcal{S},a\in\mathcal{A}} TV(\mathcal{P}(\cdot|\boldsymbol{s},a), \mathcal{P}_1(\cdot|\boldsymbol{s},a)) \leq \frac{\gamma RL}{(1-\gamma)^2}\epsilon, \\
\|V^* - V_2^*\|_\infty &\leq \frac{2\gamma R \max_{\boldsymbol{s}\in\mathcal{S},a\in\mathcal{A}} TV(\mathcal{P}(\cdot|\boldsymbol{s},a), \mathcal{P}_2(\cdot|\boldsymbol{s},a)) + (1-\gamma)\|\mathcal{R}-\mathcal{R}_2\|_\infty}{(1-\gamma)^2} \\
&\leq \frac{\gamma RL + (1-\gamma)l}{(1-\gamma)^2}\epsilon.
\end{aligned} \tag{4.8}
$$

The above theorem upper bounds the change in the optimal total accumulated reward if all time steps in the episode of game playing are perturbed. Note that in our proposed adversarial attack paradigm, we control the total number of attacked time steps in one episode. Therefore, the result from Theorem 1 on value function, which is essentially the sum of discounted rewards from all time steps, is insufficient. Accordingly, we define $\mathcal{T}$-step value function (in Appendix) to formally measure the influence of the attacker on the accumulated reward collected only from the perturbed $\mathcal{T}$ steps. Then, based on this, we can derive the following theorem.

Theorem 5. *Suppose $\mathcal{M}$, $\mathcal{M}_1$, and $\mathcal{M}_2$ satisfy the same assumptions as in Theorem 1. V^*, V_1^*, and V_2^* are optimal value functions for $\mathcal{M}$, $\mathcal{M}_1$, and $\mathcal{M}_2$, respectively. π^*, π_1^*, and π_2^* are optimal policies for $\mathcal{M}$, $\mathcal{M}_1$, and $\mathcal{M}_2$, respectively. Denote $\max(L\epsilon, l\epsilon)$ to be δ. Suppose π^* is executed on $\mathcal{M}_i$ for $\mathcal{T}$ steps, we denote $\mathcal{T}$-step value function under policy π^* as $\mathcal{V}_i^{\pi^*}(\boldsymbol{s},\mathcal{T}) = \mathbb{E}[\sum_{t=0}^{\mathcal{T}-1}\gamma^t\mathcal{R}_i(\boldsymbol{s}_t,\pi^*(\boldsymbol{s}_t))|\boldsymbol{s}_0 = \boldsymbol{s}]$. We denote the maximum possible $\mathcal{T}$-step return by $\mathcal{G}_\mathcal{T} = \max_{\boldsymbol{s}\in\mathcal{S}} \mathcal{V}^{\pi^*}(\boldsymbol{s},\mathcal{T})$. For all $\omega \geq 0$, if $\epsilon \leq \frac{1}{\max(L,l)}(\frac{\omega}{12|\mathcal{S}|\mathcal{T}\mathcal{G}_\mathcal{T}})^2$, we have $|\mathcal{V}^{\pi^*}(\boldsymbol{s},\mathcal{T}) - \mathcal{V}_i^{\pi^*}(\boldsymbol{s},\mathcal{T})| \leq \omega$, where $i\in\{1,2\}$.*

From Theorem 2, we can see that when the agent unconsciously executes the original optimal policy in the adversarially perturbed environment $\mathcal{M}_1$/$\mathcal{M}_2$, since the agent is not allowed to re-train the policy network, the fluctuation measured by $\mathcal{T}$-step value function the agent may experience, is well bounded if the perturbation the attacker imposes is small, which validates our previous argument on stealthiness of our proposed attack.

4.4 Model Poisoning Attack against DRL Interpretations

In this section, we firstly describe the threat model considered in the poisoning attack settings. Then, we present an algorithmic framework to rigorously design a model poisoning attack against DRL interpretations.

4.4.1 Threat Model

Here, we describe the threat model considered in our poisoning attack. Different from traditional data poisoning attacks where the attacker injects fake samples into the training dataset before the training process of the victim model begins, we do not assume that the attacker has full knowledge of the original training data. Instead, we assume that the attacker in our setting only has access to the pre-trained DRL model. The attacker's goal is to manipulate the parameters of the pre-trained DRL model such that the interpretation results are significantly altered. Consider the aforementioned example where due to the limited computational resources, the agent resorts to downloading the pre-trained DRL model provided by a third party. During this process, the attacker could pose potential threats to the system. To contrast with traditional data poisoning attacks [73, 74], we here call our attack the model poisoning attack since the attacker directly manipulates the pre-trained model parameters that are generated in the training phase instead of the training dataset. In practice, model poisoning attacks are more ubiquitous than data poisoning attacks as the attacker may not always have access to the training database that is typically secured by professional staff, while ordinary agent may acquire poisoned model from the attacker disguised as third-party model provider.

4.4.2 Formalization of Model Poisoning Attack

In this section, we introduce our designed model poisoning attack against DRL interpretations (i.e., MPDRLI). Note that in our model poisoning attack setting, the attacker directly manipulates the model parameters instead of the training data. However, if the performance of the re-trained DRL model is significantly degraded, the model poisoning attack can be easily detected by evaluating on the validation games, and then the re-trained DRL model will be immediately rejected by the agent. Hence, when the attacker manipulates the trained DRL model, he should guarantee that he can dramatically alter the interpretation results without significantly hurting the performance of the original DRL model. To address this challenge, we propose to design our model poisoning attack against DRL interpretations by fine-tunning the pre-trained DRL model

with the objective function that combines the ordinary loss of the original DRL model with a penalty term that involves the interpretation results. Without loss of generality, in the following, we use the DQN algorithm [59] as a representative example of DRL to present the proposed model poisoning attack (i.e., MPDRLI). Note that the proposed MPDRLI is model-agnostic and can generalize to other DRL algorithms naturally.

Note that a deep Q network estimates the Q-value function by leveraging a multi-layered neural network. The input for the network is the current state, and the output is the corresponding Q-value for each of the action. Specifically, for each given state-action pair (i.e., state $\boldsymbol{s}_t$ and action a_t), the deep Q-Network predicts the corresponding Q-Value $Q(\boldsymbol{s}_t, a_t; \boldsymbol{\Theta})$ through a forward pass, where $\boldsymbol{\Theta}$ are the parameters of the deep Q network. The value $Q(\boldsymbol{s}_t, a_t; \boldsymbol{\Theta})$ is an estimate of the expected future reward that can be obtained from $(\boldsymbol{s}_t, a_t)$. The corresponding policy for a DQN is obtained by choosing the action with the maximum Q-value for each state. The deep Q-network parameters (i.e., $\boldsymbol{\Theta}$) can be derived by minimizing the following mean-squared Bellman error

$$J(\boldsymbol{\Theta}) = \mathbb{E}[(r(\boldsymbol{s}_t, a_t) + \gamma \max_{a \in \mathcal{A}} Q(\boldsymbol{s}_{t+1}, a; \boldsymbol{\Theta}^-) - Q(\boldsymbol{s}_t, a_t; \boldsymbol{\Theta}))^2],$$

where $\boldsymbol{\Theta}^-$ represents the parameters of the target network, and the parameters $\boldsymbol{\Theta}$ of the online network are updated by sampling gradients from minibatches of past transition tuples. The above mean squared error measures the squared difference between the target Q value (i.e., $r(\boldsymbol{s}_t, a_t) + \gamma \max_{a \in \mathcal{A}} Q(\boldsymbol{s}_t, a; \boldsymbol{\Theta}^-)$) and the current Q output (i.e., $Q(\boldsymbol{s}_t, a_t; \boldsymbol{\Theta})$).

Here, we consider the case where the attacker wants to secretly alter the model parameters, such that the agent cannot figure out what features are really most important for the current action prediction with targeted interpretation method. In our model poisoning attack settings, the attacker has no knowledge of the training dataset $\mathcal{D}_{tr}$ but he can collect a substitute dataset $\mathcal{D}_{tr}^{'}$ by iteratively running the targeted model. Let $p_{t,k}(\boldsymbol{\Theta}^*)$ denote the set of pixels that had the top k highest saliency map values with interpreter $\mathcal{I}$ of the original clean DQN model (parameterized by $\boldsymbol{\Theta}^*$), for the state-action pair $(\boldsymbol{s}_t, a_t)$. Note that to avoid being detected, the attacker should maintain the performance of the retrained DRL model, while only focusing on attacking the interpretation results. In order to achieve the attack goals, based on the pre-trained model parameters $\boldsymbol{\Theta}^*$ and the substitute dataset $\mathcal{D}_{tr}^{'}$, the attacker can manipulate the original

clean DQN model as follows

$$\min_{\tilde{\boldsymbol{\Theta}}} \mathcal{L}(\tilde{\boldsymbol{\Theta}}) = \mathbb{E}[(r(\boldsymbol{s}_t, a_t) + \gamma \max_{a \in \mathcal{A}} Q(\boldsymbol{s}_{t+1}, a; \tilde{\boldsymbol{\Theta}}^-) - Q(\boldsymbol{s}_t, a_t; \tilde{\boldsymbol{\Theta}}))^2] + \lambda_3 * \frac{1}{T+1} \sum_{t=0}^{T} \sum_{j \in p_{t,k}(\boldsymbol{\Theta}^*)} \exp(\mathcal{I}(\boldsymbol{s}_t, a_t; \tilde{\boldsymbol{\Theta}})), \tag{4.9}$$

where $\boldsymbol{\Theta}^*$ is the model parameters of the original unattacked DQN model, λ_3 is a trade-off parameter, and the penalty term is designed to reduce the interpretation scores of the pixels that originally had the top k highest values. By differentiating the above loss function with respect to $\tilde{\boldsymbol{\Theta}}$, we can get the following gradient

$$\frac{\partial \mathcal{L}(\tilde{\boldsymbol{\Theta}})}{\partial \tilde{\boldsymbol{\Theta}}} = \mathbb{E}[(r + \gamma \max_{a \in \mathcal{A}} Q(\boldsymbol{s}_t, a; \tilde{\boldsymbol{\Theta}}^-) - Q(\boldsymbol{s}_t, a_t; \tilde{\boldsymbol{\Theta}})) \frac{Q(\boldsymbol{s}_t, a_t; \tilde{\boldsymbol{\Theta}})}{\partial \tilde{\boldsymbol{\Theta}}}] + \lambda_3 * \frac{1}{T+1} \sum_{t=0}^{T} \sum_{j \in p_{t,k}(\boldsymbol{\Theta}^*)} (\exp(\mathcal{I}(\boldsymbol{s}_t, a_t; \tilde{\boldsymbol{\Theta}})) * \frac{\partial \mathcal{I}(\boldsymbol{s}_t, a_t; \tilde{\boldsymbol{\Theta}})}{\partial \tilde{\boldsymbol{\Theta}}}). \tag{4.10}$$

Then, based on the above, we can re-train the DQN model by using the projected gradient descent method [59]. Note that the original parameters $\boldsymbol{\Theta}^*$ are used as the initialized parameters.

Discussion. In the above, we consider how to reduce the interpretation scores of the pixels that originally have the top k highest values. In practice, we can also make the interpretations always say that some particular region of the input (e.g., boundary or corner of the image), is important regardless of the input.

4.5 A Defense Mechanism against Malicious Attacks on DRL Interpretations

In this section, we design a novel general defense mechanism to mitigate malicious attacks on DRL interpretations. Note that, in the previous sections, we study the security vulnerabilities of DRL interpretations to two representative types of attacks against DRL interpretations: adversarial attacks and model poisoning attacks. More specifically, we first design an universal adversarial attack against DRL interpretations (i.e., UADRLI), from which the attacker can add the crafted universal perturbation uniformly to the environment states in a maximum number of steps to incur minimal damage to the agent's end goal. In addition, we also design a model poisoning attack (i.e., MPDRL), based on which the attacker can significantly alter the interpretation results while incurring minor damage to the performance of the original DRL model. With

the previously proposed malicious attack frameworks (i.e., UADRLI and MPDRL), the attacker can significantly degrade the effectiveness of the interpretation methods designed for DRL, and hence poses a serious threat to the success of the interpretation methods for DRL in practice. Thus, it is extremely essential to design effective mechanisms to defend DRL interpretations against such malicious attacks (i.e., adversarial attacks and poisoning attacks).

To address the above problem, we propose a general defense mechanism to increase the attack resistance of DRL interpretations against the proposed malicious attacks. Specifically, we propose to build an ensemble of multiple explanation methods as an simple and effective way to reduce vulnerabilities. Our approach is motivated by a key insight in machine learning: Ensemble models can reduce both bias and variance compared to applying a single model. In practice, ensemble methods have also been previously used to defend against adversarial attacks on neural network outputs [10, 75, 76], motivating the usage of an explanation ensemble to defend against malicious attacks on the generated explanations. Thus, we propose to apply the same idea to explanation methods and build an ensemble of explanation methods, which can be achieved at less computational complexity.

More specifically, we propose to calculate the average over all available explanation methods. By taking the average over all available explanation methods, we can reduce the variance of the explanation compared to using a single interpretation method. We theorize that the averaging of the diverse set of explanation methods involved in the aggregation creates smoothness. The resilience of the aggregation to malicious attacks can be understood in terms of averaging induced smoothness, and smoothness can lead to increased robustness. In addition, to get a theoretical understanding of the explanation aggregation, we propose to hypothesize the existence of a "true" explanation. In this way, we can quantify the error of an explanation method as the mean squared difference between the "true" explanation and an explanation produced by an explanation method, i.e., the mean squared error. In other words, we define the error of an explanation method as the mean squared difference between a hypothetical "true" explanation and an explanation procured by the explanation method.

Based on the above, we first discuss how to provide guarantees of robustness to adversarial attacks. Let $\mathcal{I}(\boldsymbol{s}_t, a_t; \pi)$ denote the interpreter being attacked. For the given state-action pair $(\boldsymbol{s}_t, a_t)$, we can utilize this interpreter to generate the importance vector (i.e., $\mathcal{I}(\boldsymbol{s}_t, a_t; \pi)$), which can help us to identify which features are important for the predicted action. For the motivated attackers, they can follow the above proposed malicious attacks to alter the generated interpretation results. We use $\{\tilde{\mathcal{I}}_m\}_{m=1}^{M}$ to denote the set of available clean interpreters. In the proposed adversarial attacks, the attackers aim to perturb the current state $\boldsymbol{s}_t$ to mislead the

targeted interpreter $\mathcal{I}(\cdot)$ to reduce the importance scores of the initially top-ranked features. Here, we use $\tilde{\boldsymbol{s}}_t$ to denote the perturbed version of $\boldsymbol{s}_t$. In order to defend against the adversarial attacks, we propose to average all of the generated interpretations

$$\mathcal{E}_1(\tilde{\boldsymbol{s}}_t, a_t) = \frac{\mathcal{I}(\tilde{\boldsymbol{s}}_t, a_t; \pi) + \sum_{m=1}^{M} \tilde{\mathcal{I}}_m(\tilde{\boldsymbol{s}}_t, a_t; \pi)}{M+1},$$

where M represents the number of the available clean interpreters, and $\mathcal{I}(\cdot, \cdot; \pi)$ denotes the interpreter being attacked.

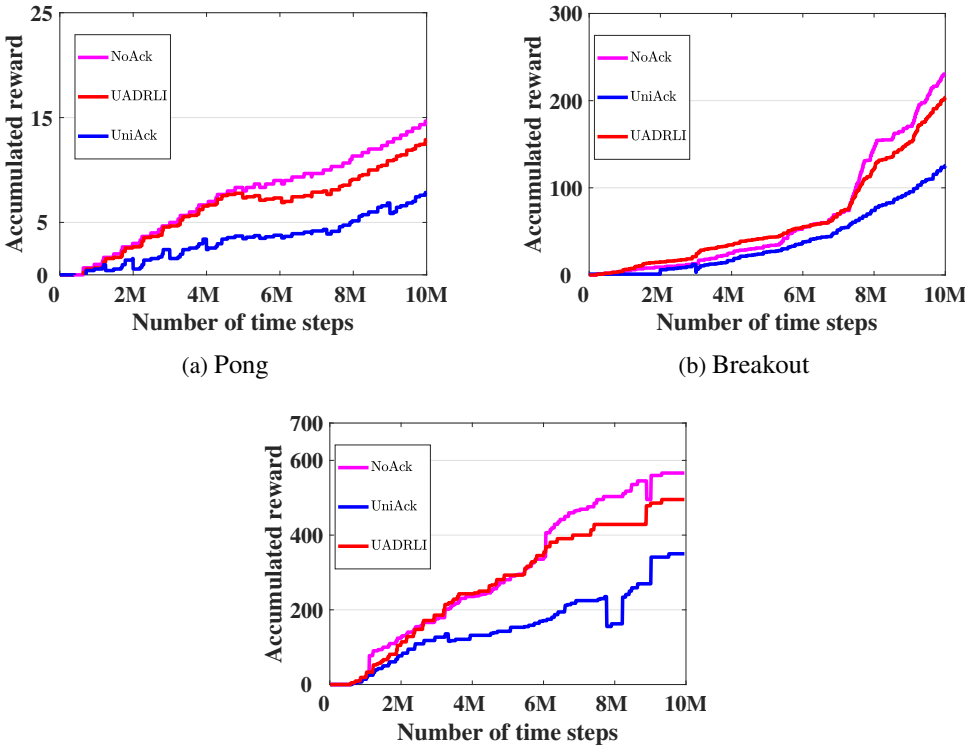
Next, we talk about how to defend model poisoning attacks, which happen at the training stage. Note that in the settings of model poisoning attacks, the attacker targets a specific interpretation method, and aims to manipulate the interpretation results (generated by the targeted interpretation method) by modifying the model parameters of the previously trained deep reinforcement learning model. In order to defend the model poisoning attacks, we also propose to average the feature importance scores that are generated from all of the available interpreters. Specifically, for each incoming testing sample, we first calculate the averaged feature importance score vector. Then, we compute the top-ranked feature overlap between the average feature importance score vector and the feature importance score vector that is derived from the targeted interpreter, which is attacked by the attacker. Similar to the above, the higher the feature intersection, the better the performance of the proposed defense mechanism. Based on this, we propose to defend model poisoning attacks by calculating the averaged feature importance scores, which is given as follows

$$\mathcal{E}_2(\boldsymbol{s}_t, a_t) = \frac{\mathcal{I}(\boldsymbol{s}_t, a_t; \tilde{\pi}) + \sum_{m=1}^{M} \mathcal{I}_m(\boldsymbol{s}_t, a_t; \tilde{\pi})}{M+1},$$

where M represents the number of the available clean interpreters, $\tilde{\pi}$ denotes the maliciously manipulated policy network, and $\mathcal{I}(\boldsymbol{s}_t, a_t; \tilde{\pi})$ is the targeted interpretation method that is attacked at the training stage.

4.6 Experiments

In this section, we firstly introduce the experimental setup in Section 4.6.1. Next, we conduct experiments to validate the effectiveness of the proposed universal adversarial attack against DRL interpretations (i.e., UADRLI) in Section 4.6.2. Then, in Section 4.6.3, we verify the effectiveness of the proposed model poisoning attack against DRL interpretations (i.e., MPDRLI). Lastly, in Section 4.6.4, we conduct experiments to evaluate the performance of the proposed



(a) Pong

(b) Breakout

(c) SpaceInvader

Figure 4.1: Performance comparison of adversarial attacks on the accumulative reward.

general defense mechanism to increase the attack resistance of DRL interpretations against the proposed malicious attacks.

4.6.1 Experimental Setup

Model setting. Our experimental implementation of the environments builds on OpenAI gym's control environments with the Atari physics simulator. In experiments, we train agents on Pong, Breakout, and SpaceInvaders by using two state-of-the-art DRL algorithms (i.e., A3C and DQN). We choose these three games because each of them poses a different set of challenges and the two adopted DRL algorithms have historically exceeded human-level performance on them. For evaluation, the game's randomness seed is reset for every episode. We also adopt two representative DRL interpreters, i.e., the Jacobian and gradient saliency.

Network architecture and parameter settings. For the adopted A3C algorithm, all of the Atari agents have the same recurrent architecture. The input at each time step is a pre-processed version of the current frame, and the preprocessing operations include gray-scaling,

Table 4.1: The setting of parameters.

Parameter	Value	Parameter	Value
learning rate	1e-4	λ_1	1.0
discount factor (γ)	0.99	k	706
λ_3	1.0	-	-

down-sampling by a factor of 2, cropping the game space to an 80×80 square and normalizing the values to $[0, 1]$. This input is processed by 4 convolutional layers (each with 32 filters, kernel sizes of 3, strides of 2, and paddings of 1), followed by an LSTM layer with 256 hidden units and a fully connected layer with $|\mathcal{A}| + 1$, where $|\mathcal{A}|$ denotes the dimension of action space. For the adopted DQN algorithm, the network architecture is a convolutional neural network with 3 convolution layers and a fully-connected hidden layer. Specifically, the first hidden layer convolves 32 filters of 8×8 with stride 4 with the input image and applies a rectifier nonlinearity. The second hidden layer convolves 64 filters of 4×4 with stride 2, again followed by a rectifier nonlinearity. The third hidden layer convolves 64 filters of 3×3 with stride 1 followed by a rectifier. The final fully-connected layer consists of 512 rectifier units. Here, the input RGB frame (i.e., the observed state) is rescaled to 84×84. The output layer is a fully-connected linear layer with a single output for each valid candidate action. The setting of parameters is given in Table 4.1.

Baselines. For the proposed universal attack, in experiments, we adopt two baselines: Firstly, we adopt the uniform adversarial attack as the baseline, denoted as **UniAck**, which is a direct extension of the traditional adversarial attacks on DRL. In UniAck, we apply the generated universal perturbation to the observed state at each time step. Additionally, we also compare the agent's performance under our proposed adversarial attacks with that under no adversarial attack, denoted as **NoAck**. For the proposed model poisoning attack, since there is no existing work addressing the vulnerability of DRL to poisoning attacks, we adopt the no model poisoning attack baseline (dubbed as **NoPAck**).

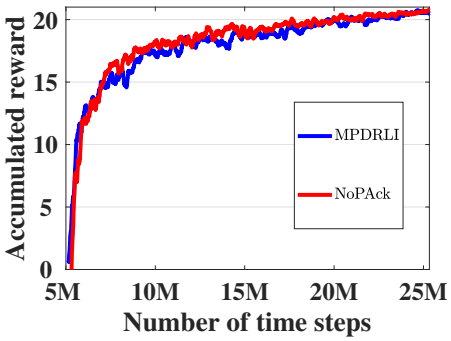
4.6.2 Experiments for Adversarial Attack

In this section, we evaluate the performance of the proposed universal adversarial attack (i.e., UADRLI). Unless otherwise specified, in this experiment, we adopt the Jacobian saliency and restrict the attacker to only manipulating the pixels at the top-left corner of the input image, and set the size of the attacked image area as 40×40. Additionally, given an episode that consists

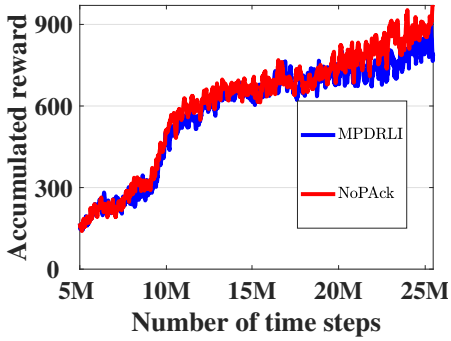
of an alternating sequence of state-action pairs, for the proposed UADRLI, we only attack 10% of these state-action pairs that have the lowest Q value variance. In contrast, for the baseline UniAck, we attack 10% of these state-action pairs that have the largest Q value variance.

Performance of adversarial attacks on the discounted accumulative reward. Next, we compare the performance of the proposed UADRLI with that of the two baselines by averaging the total reward accumulated by the target agent. Here, for the proposed UADRLI, the universal perturbation $\boldsymbol{\delta}$ is crafted with $\epsilon = 0.12$. For an episode of game playing, we only attack 10% of the state-action pairs that have the lowest Q value variance. In contrast, for the adopted baseline UniAck, we attack 10% of the state-action pairs that have the largest Q value variance. The experimental results on the three adopted games are shown in Figure 4.1, where the y-axis is the accumulated reward and the x-axis is the number of time steps. The reference line in the figure is the purple line which corresponds to the reward function under no attack (NoAck). From this figure, we can see that the proposed UADRLI can reach the similar effect of the original unattacked DRL model. In contrast, for the adopted baseline UniAck that attacks the time steps where the variance of the corresponding states are high, it suffers from the most severe reduction in accumulated reward, which is also in accordance to the conclusion of Lemma 5 (in Appendix) that attacking the states with low variance incurs low decrease in the accumulated reward. In sum, regardless of which game the agent plays, the proposed UADRLI indeed incurs minor decrease in the policy's performance.

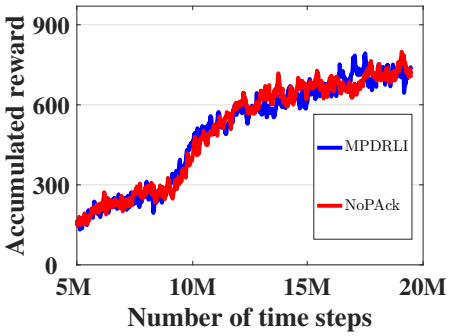
Visualization. Next, we visually demonstrate the effectiveness of the proposed UADRLI. To better visualize the experimental results, in this experiment, we set the value of ϵ as 0.12, and set the size of the attacked image region as 20×20. In practice, we can set ϵ as a much smaller value to make the crafted universal perturbation $\boldsymbol{\delta}$ more imperceptible. In Figure 4.2, we plot the visualization results on the Pong game. In this figure, the leaftmost (i.e., Figure 4.2a) is the original input image, the second one (i.e., Figure 4.2b) is the adversarial image that is derived by adding the crafted universal perturbation $\boldsymbol{\delta}$ to its original unattacked state (i.e, Figure 4.2a), and the rightmost (i.e, Figure 4.2c) highlights the most important features that are identified by the adopted Jocabian interpreter. By adding the universal perturbation, the trained agent, who should have taken the "down" action, take the "up" action instead. And the added perturbations at the top-left corner are the the real reason for the wrong action prediction. However, from Figure 4.2c, we can observe that the crafted universal perturbation can successfully fool the adopted interpreter. That is to say, the adopted interpreter cannot identify the attacked image region. These results show that the proposed UADRLI not only leads the trained agent to make wrong action judgement but also can avoid being detected.



(a) Pong

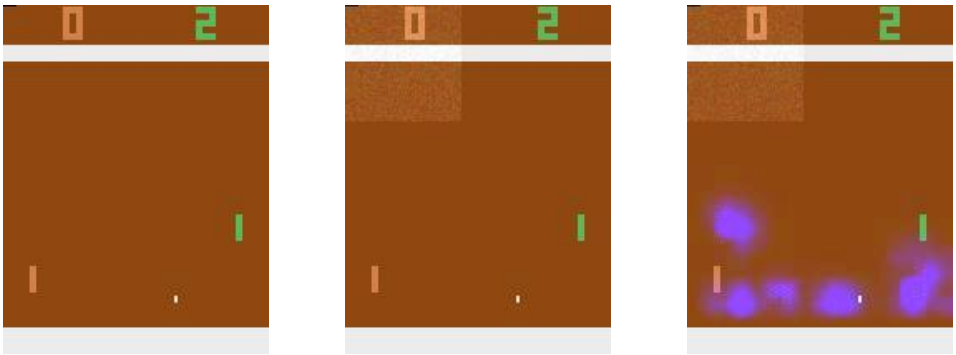


(b) Breakout



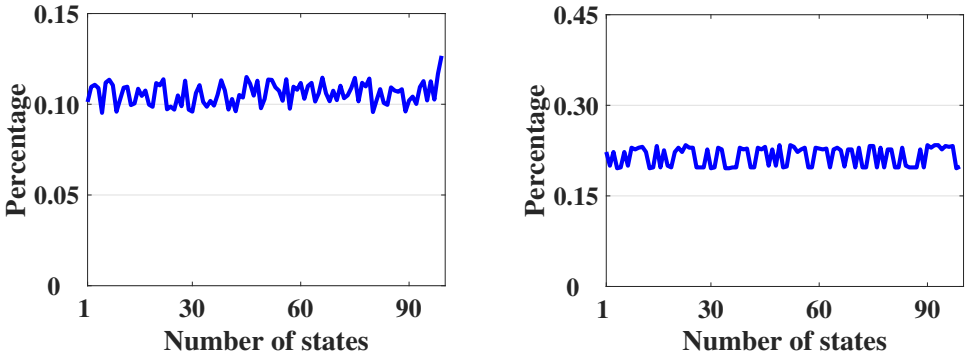
(c) SpaceInvader

Figure 4.3: Performance of the proposed model poisoning attack on the accumulative reward.



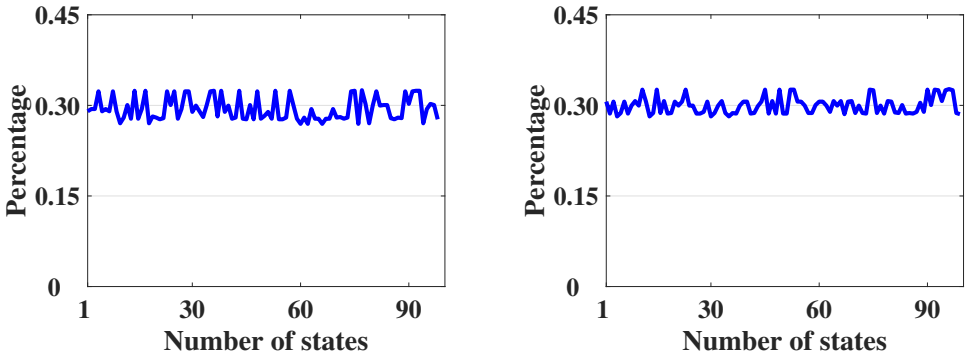
(a) (b) (c)

Figure 4.2: Visualization results for the Pong game.



(a) Pong (b) Breakout

Figure 4.4: Percentage of identified features when $k = 706$.



(a) Pong (b) Breakout

Figure 4.5: Percentage of identified features when $k = 1,058$.

4.6.3 Experiments for Model Poisoning Attack

In this section, we evaluate the performance of the proposed model poisoning attack (i.e., MPDRLI). The adopted DRL algorithm is DQN, and the adopted interpreter is the gradient saliency. In the following experiment, unless otherwise specified, we set $\lambda_3 = 1.0$ and $k = 706$, and there are $84 * 84$ features in total.

Performance on the accumulated discounted reward. First of all, we compare the performance of the proposed MPDRLI with that of the adopted baseline (i.e., NoPAck) on the accumulated reward. The experimental results are reported in Figure 4.3, where x-axis denotes the number of time steps and y-axis represents the accumulated reward. From this figure, we can see that the proposed MPDRLI achieves similar performance as that of the adopted no attack baseline (i.e., NoPAck). These experimental results verify that the proposed model poisoning

attack incurs minor decrease to the performance of the original DRL model. In this way, the proposed poisoning attack can remain undetected (i.e., stealthy). In practice, if the performance of the poisoned DRL model is significantly decreased, the agent can easily detect the launched attacks by just checking the accumulated reward.

Performance on altering the interpretation results. We then evaluate the effectiveness of the proposed MPDRLI in terms of altering the interpretation results. In this experiment, we first let the agent play a game with the original DRL model, and then select a sequence of state-action pairs from this entire game episode. Next, for each selected state-action pair, we first use the adopted interpreter to identify the k highest ranked features that are crucial for explaining the prediction decision made by the original DRL model. Then, for this state-action pair, we select the k highest ranked features that explain the decision made by the poisoned DRL model. After that, we count the number of the features in the intersection between the two selected feature subsets. The lower the number of features in the intersection, the better the performance of the proposed MPDRLI. Based on this count, we can calculate the percentage of features in the intersection over the number of features originally identified by the interpreter. The averaged experimental results are reported in Figure 4.4. Here, we vary the number of the selected state-action pairs from 1 to 99. Take Figure 4.4a as an example. From this figure, we can see that the percentage of the features in the intersection is only around 0.10. In other words, the interpretation results generated from the poisoned DRL model are significantly different from that of the original clean model. Hence, the proposed MPDRLI can guarantee that the top-ranked features cannot be identified by the agent. In Figure 4.5, we also show the experimental results when the value of k is set as $1,058$, which means that the attacker attacks 15% of all the features that have the largest feature importance scores. From this figure, we can also derive the same observation that the interpretation results generated from the poisoned DRL model are significantly different from that of the original clean DRL model.

4.6.4 Experiments for the Proposed Defense Mechanism

In this section, we will conduct experiments to evaluate the performance of the proposed defense mechanism. Note that in the previous sections, we studied the vulnerability of deep reinforcement learning interpretations by presenting the algorithmic frameworks to rigorously formulate the malicious attacks. The proposed malicious attacks aim to reduce the importance scores of the features that originally had the top k highest values.

	Number of time steps					
Datasets	10	20	30	40	30	40
Breakout	0.205	0.212	0.207	0.215	0.245	0.205
Breakout (defense)	0.627	0.632	0.687	0.637	0.621	0.678

Table 4.2: Percentage of identified features on the game of Breakout.

We here conduct experiments to evaluate the performance of the proposed defense mechanism against the adversarial attacks. Specifically, in this experiment, we first generate the adversarial instance for each of the instances in the test set based on the proposed adversarial attack strategy described in the previous section. Then, by using all of the available interpreters, we calculate the averaged feature importance score vector for each adversarial instance. Here the derived feature importance score vectors are derived based on the proposed defense mechanism. After that, we can compute the feature overlap based on the averaged feature importance score vector and the original feature importance score vector for the original clean instance. Finally, we can calculate the percentage of features in the intersection over the number of features originally identified by the interpreter. The larger the feature overlap, the better the performance of the proposed defense mechanism. In addition, to estimate the performance of the proposed defense mechanism against the model poisoning attacks, for each incoming testing sample, we first calculate the averaged feature importance score vector. Then, we compute the top-ranked feature overlap between the average feature importance score vector and the feature importance score vector that is derived from the targeted interpreter. Similar to the above, the higher the feature intersection, the better the performance of the proposed defense mechanism. The averaged experimental results on the game of Breakout are reported in Figure 4.2. In this experiment, we trained the agents on the game of Breakout, and varied the number of the time steps from 10 to 40. From the reported experimental results this figure, we can see that the interpretation results (i.e., the feature overlap) generated from the defense mechanism are significantly larger than that of the attacked interpreter.

4.7 Related Work

Adversarial attacks against deep reinforcement learning. Recent studies [77, 78, 79, 80, 81] show that DRL algorithms are unavoidably susceptible to adversarial perturbations. [77] makes use of white-box assumptions and proposes an attack method where the attacker attacks every time step by applying the FGSM. [78] designs a targeted controlling attack where the

attacker can manipulate the policy by adding the imperceptible noise to the observations of the environment. [79] proposes adversarial attacks that lead the agent into increased probability of taking worst possible actions. [80] verifies the transferability of adversarial examples across different DQN models. [81] unveils how little it takes to deceive an DRL policy by considering three restrictive settings. However, these methods only focus on attacking specific states using traditional per-instance techniques, and ignore the end goal of the entire DRL task. In contrast, [82, 83] consider how to significantly deteriorate the agent's end reward. However, the agent can easily identify the adversarial attack by simply comparing the consequent end goal with his own desired one. Additionally, all of the above works only focus on how to craft state-dependent perturbations, which is computationally intractable in long state sequences. They also make an implicit assumption that the attacker has the ability of manipulating the whole input data. Lastly, they do not study the vulnerability of DRL interpretations to the security threats.

Interpretation models for deep reinforcement learning. Based on the interpretation stages, existing DRL interpretation works can be generally divided into the following two categories: intrinsic and post-hoc. The latter case does not require modifying the model architectures or parameters, thereby leading to higher prediction accuracy [84]. Motivated by this, considerable works [59, 62, 63, 61, 64, 65, 66] have been proposed to provide post-hoc explanations for explaining the model's output for a given input. For example, [64] takes a closer look at a slightly modified version of Grad-CAM in the context of deep RL on Atari games. However, all of these interpretation works assume a secure and reliable environment, and do not consider the vulnerability of DRL interpretations to the malicious attacks.

Adversarial attacks against deep learning interpretations. Very recently, some works [67, 68, 69, 70, 71] are beginning to study the vulnerability of the interpretation methods for deep neural networks. For example, [69] demonstrates that explanation maps can be sensitive to small perturbations in the image. Their results can be thought of as untargeted manipulations, i.e., perturbations to the image which lead to an unstructured change in the explanation map. However, these works only focus on how to design adversarial attacks against supervised deep neural network interpretations, and cannot be directly applied to DRL due to its unique characteristics (e.g., the sequentiality of decision-making process and the end-goal oriented property). Furthermore, these works only consider the adversarial attack, and do not consider the poisoning attack.

4.8 Conclusions

To the best of our knowledge, we are the first to study the vulnerability of DRL interpretations to the malicious attacks. More specifically, in this chapter, we firstly present an universal adversarial attack against DRL interpretations (i.e., UADRLI), from which the attacker can add the crafted universal perturbation uniformly to the environment states in a maximum number of steps to incur minimal damage to the agent's end goal. Then, we design a model poisoning attack against DRL interpretations (i.e., MPDRLI), based on which the attacker can significantly alter the interpretation results while incurring minor damage to the performance of the original DRL model. To enhance the robustness of DRL interpretations against malicious attacks, we also propose a general defense mechanism to increase the attack resistance of DRL interpretations against the proposed malicious attacks. Both theoretical analysis and extensive experimental results are provided to demonstrate the effectiveness of our proposed approaches.

Chapter 5

Robust and Automatic Model Explanations

5.1 Introduction

Recently, interpreting and understanding the behaviors of black-box machine learning (ML) models has drawn significant attention [85, 86, 87, 88, 89, 90, 91, 90, 6, 92]. The most commonly-used explanation method is to explain an ML model's predictions in terms of the input features (e.g., pixels and word-vectors) [10, 12, 13, 14, 15, 26, 93, 94]. However, these feature-based interpretations suffer from several drawbacks [85]. For example, [86] demonstrates that given identical feature-based explanations, human can confidently find evidence for completely contradicting conclusions. In addition, the feature-based explanation methods are not necessarily the most intuitive explanations for human understanding, especially when using low-level features (e.g., the raw pixels). In contrast, human reasoning often comprises "concept-based thinking" by extracting similarities from numerous samples and grouping them semantically based on their resemblance [95]. As a consequence, recent research has focused on designing concept-based explanation methods to interpret how ML models use high-level human-understandable concepts in arriving at decisions [86, 85, 96, 97, 98, 87, 95, 99].

However, an obstacle to the large-scale adoption of these concept-based explanation methods is that they require significant human effort and resource expenditure. The reason is that existing concept-based explanation methods implicitly follow a two-stage procedure with manual intervention. Specifically, they first need human to manually define concepts by using a set of input examples for the ML model under inspection [85, 86, 87], and then manually compute

the importance score of each pre-defined concept in a post-hoc way. For example, to define the concept of "curly", [85] needs a human subject to go over all the given images of this concept and extract meaningful segmentations. Then, [85] manually computes each extracted concept's importance score via the directional derivative method [86]. However, identifying human-interpretable concepts and checking for the semantic meaningfulness require a large effort from human experts due to manual annotations and computation. Thus, how to automatically provide concept-based explanations without human intervention still remains a fundamental challenge.

Our goal in this chapter is to automatically generate the intrinsic concept-based explanations from the input data without human intervention. To achieve this goal, we propose to inject the concept-based explanations into the learning loop: whenever asking the user to label an incoming sample, the model can simultaneously provide the predicted label for this sample and the corresponding concept-based explanations on interpreting this predicted label. However, the challenge here is how to define the units of concept-based explanations from the learning network structure considering that they are very subjective. If we directly follow existing concept-based works [86, 85, 96, 97, 98, 87, 95, 99], the human has to be involved in this laborious tuning process. The reason is that ML models usually do not comprehend the way humans do and cannot guarantee that the extracted concepts are semantically meaningful, which also violates the fidelity of concept-based explanations. So the method proposed in [85] still needs human involvement (e.g., removing outliers segments of each concept). In addition, the number of the interpretability units also determines the construction of the self-explanatory network architecture based on the desired properties. Hence, instead of manual tuning, we also need to address how to automatically learn the optimal number of interpretability units.

Furthermore, the concerns regarding the reliability of explanations still exist [100]. In practice, motivated attackers could generate imperceptible adversarial perturbations to change the interpretability of the input data while preserving the predicted results [101, 69, 102, 103]. This lack of robustness is problematic in real-world applications where adversarially manipulated explanations could impair safety and trustworthiness. For instance, given a traffic sign classification, a prediction classifying an input as a stop sign with the explanation that the background contains a river is unlikely to be trusted by the users. Although there are some works [104, 100, 105, 106, 107, 108] exploring the robustness of model explanations, they cannot be certified, which means that no provable guarantees can be given to verify their robustness. In practice, these uncertified methods become vulnerable under stronger adversarial attacks. Thus, it is also of great importance to rigorously guarantee robustness of the generated explanations. With such certifiable robustness guarantees for the generated explanations, we need not worry

about an adversary with a stronger optimizer, or a more clever algorithm for choosing adversarial perturbations.

In order to tackle the above challenges, in this chapter, we design a novel automatic and robust model interpretation method (AutoRMI), a self-explanatory model that can not only automatically provide the concept-based explanations via units that are more understandable to humans than individual features (e.g., pixels) but also provide certified robustness guarantees for the generated explanations. In our proposed method, given that defining the units of concept-based explanations is very subjective, we first propose an interpretability regularization term that guides the model to extract the prototype-based concepts from the training data during the training process. More specifically, each prototype-based concept in our setting is a representative instance that best presents a possibly target set and summarizes the underlying data pattern. Since these prototype-based concepts are extracted in a way that they can represent a set of particular targets in the training data, we can guarantee that these extracted concepts have meaningful and relevant information. Hence, we can release the burden of human from the multifarious manual engagement process. Additionally, to reduce the susceptibility of the generated explanations to adversarial attacks, we also design a novel interval bound propagation based regularization term, which is a bounding technique derived from interval arithmetic [109, 110, 111] and is an incomplete method for training verifiably robust models. Specifically, this bounding based regularization term minimizes an upper bound on the maximum difference between any pair of explanation results when the input can be perturbed within a norm-bounded ball, and is computationally efficient since its computational cost is comparable to two forward passes through the network. Extensive experiments on real-world datasets demonstrate the effectiveness of the proposed interpretation method.

5.2 Methodology

Note that our goal is to design a novel self-explaining framework, which can not only automatically provide the concept-based explanations without requiring any human intervention but also provide certified robustness guarantees for the generated explanations. However, as aforementioned, defining the units of concept-based explanations is very challenging since they are very subjective. To tackle this challenge, we propose to extract the prototype-based concepts in the training data to guide the model to explain predictions. These learned prototype-based concepts are the representative patterns that describe influential data structures in latent representations. Specifically, we first build an autoencoder component to find the smallest possible

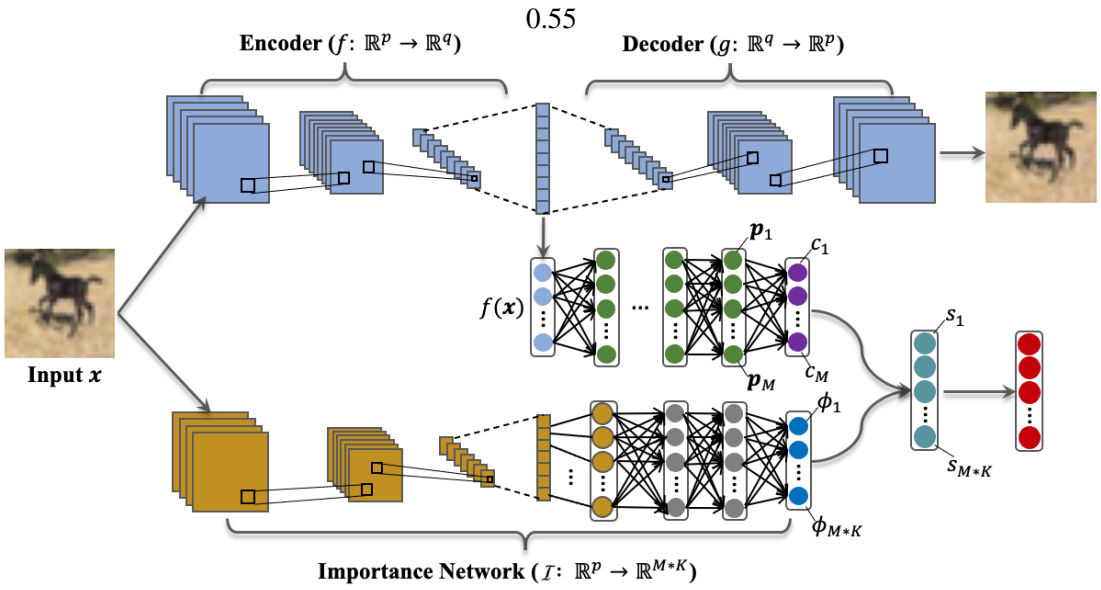


Figure 5.1: Model structure of the proposed method.

representation of data that it can store, and then design a novel interpretation regularizer to extract the prototype-based concepts during training. After that, in order to promote certified robust interpretability, we propose a novel bounding based regularization term. Below, we first give an overview on the model architecture of the proposed method, and then detail the learning objective.

Overview. Formally, we denote the training dataset by $\mathcal{X} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in \{1, \cdots, K\}$. The neural architecture of the proposed AutoRMI is presented in Figure 5.1. The proposed neural architecture includes an autoencoder network, a concept network, and an importance network. Specifically, the autoencoder network learns a lower-dimension latent representation of the data with an encoder network, $f : \mathbb{R}^p \rightarrow \mathbb{R}^q$. By using the decoder function ($g : \mathbb{R}^q \rightarrow \mathbb{R}^p$), we can project the latent space back to the original dimension. Then, we can pass the learned latent representation (i.e., $f(\boldsymbol{x})$) to the concept network, i.e., $h : \mathbb{R}^q \rightarrow \mathbb{R}^K$. The concept network first uses several fully connected layers over the latent space to learn M prototype-based concepts, i.e., $\{\boldsymbol{p}_m \in \mathbb{R}^q\}_{m=1}^M$. These prototype-based concepts ($\{\boldsymbol{p}_m \in \mathbb{R}^q\}_{m=1}^M$) can provide insight into the representative patterns across the training data that are utilized by the model for predictions. By using the decoder g, we can decode the learned prototype-based concepts to examine what the model has learned. After that, for each $\boldsymbol{p}_m$, the similarity layer computes its distance from the learned latent representation (i.e., $f(\boldsymbol{x})$)

as $c_m = ||f(\boldsymbol{x}) - \boldsymbol{p}_m||_2^2$. The smaller the distance value is, the more similar $f(\boldsymbol{x})$ and $\boldsymbol{p}_m$ are. The importance network (i.e., $\mathcal{I} : \mathbb{R}^q \rightarrow \mathbb{R}^{M*K}$) is trained to quantify the importance scores (i.e., $\boldsymbol{\Phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \cdots, \phi_{M*K}(\boldsymbol{x})]$) of different prototype-based concepts for the predicted result $y(\boldsymbol{x}) = h(f(\boldsymbol{x}))$. Finally, the similarity vector (i.e., $\boldsymbol{c}(\boldsymbol{x})$) and the importance vector (i.e., $\boldsymbol{\Phi}(\boldsymbol{x})$) are aggregated for classification. We use $y(\boldsymbol{x}) = h(f(\boldsymbol{x}))$ to denote the predicted classification result for sample $\boldsymbol{x}$.

The reconstruction error. Note that the autoencoder network here performs data compression and compresses high dimensional data into latent representations via extracting the most prominent features of the original data, and consists of an encoder ($f : \mathbb{R}^p \rightarrow \mathbb{R}^q$) and a decoder ($g : \mathbb{R}^q \rightarrow \mathbb{R}^p$). The input of the encoder is a data sample and its output is the smallest possible latent representation of that sample. The decoder ($g : \mathbb{R}^q \rightarrow \mathbb{R}^p$) takes the latent representation and can project it back to reconstruct the original sample. Here, we use $\tilde{\boldsymbol{x}}_i$ to denote the reconstruction of the original sample $\boldsymbol{x}_i$. The reconstruction loss term for the autoencoder network can be computed as the following sum of the difference between the original input and the consequent reconstruction

$$\mathcal{L}_1(\{\boldsymbol{x}_i\}_{i=1}^N, \{\tilde{\boldsymbol{x}}_i\}_{i=1}^N) = \sum_{i=1}^{N} ||\boldsymbol{x}_i - g(f(\boldsymbol{x}_i))||_2^2, \tag{5.1}$$

where $\tilde{\boldsymbol{x}}_i$ is the reconstruction of $\boldsymbol{x}_i$. In the above, $g(\cdot)$ and $f(\cdot)$ denote the decoder and encoder, respectively.

The interpretability regularization term. Note that the concept network (i.e., $h : \mathbb{R}^q \rightarrow \mathbb{R}^K$) first learns M concept vectors $\{\boldsymbol{p}_m \in \mathbb{R}^q\}_{m=1}^M$ in the latent space during the training process and then generates a probability distribution over the K classes for each test sample. Due to the subjectivity of defining the units of concept-based explanations, we propose to learn a set of prototype-based concepts (i.e., $\{\boldsymbol{p}_m\}_{m=1}^M$) during model training. These prototype-based concepts are extracted in a way that they can best represent some specific target sets and capture the influential data structures in latent representations, which ensures the semantic meaningfulness of these extracted concepts. With these extracted concepts, we can gain direct insight into representative patterns that are used by the model for classification tasks. For the encoded input $f(\boldsymbol{x}_i)$, the similarity layer computes its squared ℓ_2 distance from each of the prototype-based concepts as $c(\boldsymbol{x}_i) = [c_1 = ||f(\boldsymbol{x}_i) - \boldsymbol{p}_1||_2, \cdots, c_M = ||f(\boldsymbol{x}_i) - \boldsymbol{p}_M||_2]^T$. To enable the model to learn representative patterns from the original input data, we formulate the

following interpretability regularization loss term

$$\mathcal{L}_2(\{\boldsymbol{p}_m\}_{m=1}^M, \{\boldsymbol{x}_i\}_{i=1}^N) = \frac{1}{M}\sum_{m=1}^{M}\min_{i\in[1,N]}||\boldsymbol{p}_m - f(\boldsymbol{x}_i)||_2^2 + \frac{1}{N}\sum_{i=1}^{N}\min_{m\in[1,M]}||f(\boldsymbol{x}_i) - \boldsymbol{p}_m||_2^2 + \frac{2}{M(M-1)}\sum_{m=1}^{M}\sum_{\tilde{m}=m+1}^{M}\max(0, d_{min} - ||\boldsymbol{p}_m - \boldsymbol{p}_{\tilde{m}}||_2)^2, \tag{5.2}$$

where d_{min} is a threshold that classifies whether two prototype-based concepts are close or not. The minimization of the first loss term (i.e., $\frac{1}{M}\sum_{m=1}^{M}\min_{i\in[1,N]}||\boldsymbol{p}_m - f(\boldsymbol{x}_i)||_2^2$) enforces each prototype-based concept $\boldsymbol{p}_m$ to be as close as possible to at least one of the training examples in the latent space, which will push each prototype-based concept to learn one of the encoded training examples. The minimization of the second loss term is utilized to enforce the encoded training examples in the latent space to be close to one of the concepts, such that the training examples will be clustered around prototypes in the latent space. The third term is a diversity regularization term that exerts a larger penalty on smaller pairwise distances between the prototype-based concepts. By keeping the prototypes distributed in the latent space, it also helps produce a sparser similarity vector.

The misclassification error. Note that for $\boldsymbol{x}_i$, its learned lower-dimension representation in the latent space $f(\boldsymbol{x}_i)$ is passed to the concept network (i.e., $h : \mathbb{R}^q \rightarrow \mathbb{R}^K$) for classification. Specifically, for $\boldsymbol{x}_i$, its learned importance vector (i.e., $\boldsymbol{\Phi}(\boldsymbol{x}_i) = [\phi_1(\boldsymbol{x}_i), \cdots, \phi_{M*K}(\boldsymbol{x}_i)]$) and similarity vector (i.e., $\boldsymbol{c}(\boldsymbol{x})$) are aggregated for classification. Let $h_k(f(\boldsymbol{x}_i))$ denote the probability of $\boldsymbol{x}_i$ belonging to class $k \in [K]$. The cross-entropy loss on the training data (i.e., $\{\boldsymbol{x}_i\}_{i=1}^N$) is given as follows

$$\mathcal{L}_3 = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} L_y(h_k(f(\boldsymbol{x}_i)), y_i), \tag{5.3}$$

where L_y is used for penalizing the misclassification.

The certified robust interpretability regularizer. Here, our goal is to provide certified robustness guarantees for the generated explanations (i.e., the importance vector $\boldsymbol{\Phi}(\boldsymbol{x})$). To achieve this goal, we propose to use interval bound propagation to minimize an upper bound on the worst-case loss that any adversarial attack can perturb the generated explanations. Note that the input to the importance network (i.e., $\mathcal{I} : \mathbb{R}^q \rightarrow \mathbb{R}^{M*K}$) is denoted $\boldsymbol{x}$ and its output is an importance vector which provides concepts' importance scores. For clarity of presentation, we assume that the importance network is defined by a sequence of transformations h^t for each of

its t layers. We use $z^{(t)}$ to denote the output of layer t, where n_t is the number of units in the t-th layer and $z^{(0)}$ stands for the input. Specifically, the network computes

$$z^t = h^{t-1}(z^{(t-1)}), \forall t = 1, \cdots, T, \tag{5.4}$$

where $z^t \in \mathbb{R}^{M*K}$. Here, we consider the top-k robustness, which requires that the set of concepts with the k highest importance scores remains invariant over small-norm adversarial perturbations. In practice, the top-k attack [69, 102, 112, 113, 114] seeks to perturb the concept importance map by decreasing the relative importance of the k initially most important concepts. Let $[D]$ and $S_{\boldsymbol{x},k}$ denote the index set of the concepts and the set of concepts that had the top k highest importance scores for sample $\boldsymbol{x}$, respectively. Let $\bar{S}_{\boldsymbol{x},k} = [D] - S_{\boldsymbol{x},k}$. To produce a certification for the generated explanations (i.e., the importance scores $\boldsymbol{\Phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \cdots, \phi_{M*K}(\boldsymbol{x})]$) of sample $\boldsymbol{x}$, we aim to verify the following condition is true

$$\min_{j \in S_{z^{(0)},k}} \underline{\phi}_j(z^{(0)}) - \max_{\bar{j} \in \bar{S}_{z^{(0)},k}} \bar{\phi}_{\bar{j}}(z^{(0)}) \geq 0, \forall z^{(0)} \in \mathbb{B}(\boldsymbol{x}),$$

where $z^{(0)} = \boldsymbol{x}$. Here, $\underline{\phi}_j(\cdot)$ and $\bar{\phi}_j(\cdot)$ denotes the upper and lower bound, respectively. $\mathbb{B}(\boldsymbol{x}) = \{\boldsymbol{x}' | ||\boldsymbol{x} - \boldsymbol{x}'||_\infty \leq \epsilon\}$ is the constraint set over which the adversarial input ranges. Next, we describe how to produce certificates using interval bound propagation as in [115]. Specifically, we propose to bound the activation z_t of each layer by an axis-aligned bounding box (i.e., $\underline{z}_t(\epsilon) \leq z_t \leq \bar{z}_t(\epsilon)$[1]) using interval arithmetic. For each coordinate $z_{t,i}$ of z_t, we have

$$\underline{z}_{t,i}(\epsilon) = \min_{\underline{z}_{t-1}(\epsilon) \leq z_{t-1} \leq \bar{z}_{t-1}(\epsilon)} e_i^T h_t(z_{t-1}), \tag{5.5}$$

$$\bar{z}_{t,i}(\epsilon) = \max_{\underline{z}_{t-1}(\epsilon) \leq z_{t-1} \leq \bar{z}_{t-1}(\epsilon)} e_i^T h_t(z_{t-1}), \tag{5.6}$$

where $\underline{z}_0(\epsilon) = \boldsymbol{x} - \epsilon \mathbf{1}$, $\bar{z}_0(\epsilon) = \boldsymbol{x} + \epsilon \mathbf{1}$, and e_i is a one-hot vector with 1 in the i-th position. The above optimization problems can be solved quickly and in closed form for affine layers and monotonic activation functions. Specifically, for the affine layers (e.g., fully connected layers, convolutions) that can be represented in the form $z^t = h^{t-1}(z^{(t-1)}) = W^t z^{t-1} + b^{(t)}$, we can get an outer approximation of the tractable interval range of activations by the next layer z^t using the following formula

$$\begin{aligned} \bar{z}^{(t)} &= W^{(t)} \frac{\bar{z}^{(t-1)} + \underline{z}^{t-1}}{2} + |W^{(t)}| \frac{\bar{z}^{(t-1)} - \underline{z}^{t-1}}{2} + b^{(t)}, \\ \underline{z}^{(t)} &= W^{(t)} \frac{\bar{z}^{(t-1)} + \underline{z}^{t-1}}{2} - |W^{(t)}| \frac{\bar{z}^{(t-1)} - \underline{z}^{t-1}}{2} + b^{(t)}. \end{aligned} \tag{5.7}$$

[1] For simplicity, we abuse the notation $\leq$ to mean that all coordinates from the left-hand side need to be smaller than the corresponding coordinates from the right-hand side.

Here, $\bar{z}^{(t-1)}$ denotes the upper bound of each interval, $\underline{z}^{(t-1)}$ the lower bound, and $|W^{(t)}|$ the element-wise absolute value. In the similar way, if $h^{(t)}(z^{(t-1)})$ is an element-wise monotonic activation (e.g., a ReLU), then we can calculate the outer approximation of the reachable interval range of the next layer using the following formulas

$$\bar{z}^{(t)} = h^{(t)}(\bar{z}^{(t-1)}), \quad \underline{z}^{(t)} = h^{(t)}(\underline{z}^{(t-1)}). \tag{5.8}$$

Then, by iteratively applying the above rules, we can propagate intervals through the network and eventually get $\bar{z}^{(T)}$ and $\underline{z}^{(T)}$ (i.e., $\bar{\boldsymbol{\Phi}}(\boldsymbol{x})$ and $\underline{\boldsymbol{\Phi}}(\boldsymbol{x})$). A certificate can then be given if we can show that the above verification condition is always true for outputs in the range $\bar{z}^{(T)}$ and $\underline{z}^{(T)}$. Based on this, we propose to minimize the following robustness loss during the training process to provide certified robustness guarantees for the generated explanations

$$\mathcal{L}_4 = \frac{1}{N}\sum_{i=1}^{N} \max(\max_{\bar{j}\in\tilde{S}_{\boldsymbol{x}_i,k}} \bar{\phi}_{\bar{j}}(\boldsymbol{x}_i) - \min_{j\in S_{\boldsymbol{x}_i,k}} \underline{\phi}_j(\boldsymbol{x}_i), 0), \tag{5.9}$$

where $\bar{S}_{\boldsymbol{x}_i,k} = [D] - S_{\boldsymbol{x}_i,k}$.

Full objective. To summarize, the overall loss that we are minimizing is

$$\begin{aligned}\mathcal{L} = &\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} L_y(h_k(f(\boldsymbol{x}_i)), y_i) + \lambda_1 \sum_{i=1}^{N} ||\boldsymbol{x}_i - g(f(\boldsymbol{x}_i))||_2^2 \\ &+ \frac{\lambda_2}{M}\sum_{m=1}^{M} \min_{i\in[1,N]} ||\boldsymbol{p}_m - f(\boldsymbol{x}_i)||_2^2 + \frac{\lambda_3}{N}\sum_{i=1}^{N} \min_{m\in[1,M]} ||f(\boldsymbol{x}_i) - \\ &\boldsymbol{p}_m||_2^2 + \frac{2\lambda_4}{M(M-1)}\sum_{m=1}^{M}\sum_{\tilde{m}=m+1}^{M} \max(0, d_{min} - ||\boldsymbol{p}_m - \boldsymbol{p}_{\tilde{m}}||_2)^2 \\ &+ \lambda_5 \frac{1}{N}\sum_{i=1}^{N} \max(\max_{\bar{j}\in\tilde{S}_{\boldsymbol{x}_i,k}} \bar{\phi}_{\bar{j}}(\boldsymbol{x}_i) - \min_{j\in S_{\boldsymbol{x}_i,k}} \underline{\phi}_j(\boldsymbol{x}_i), 0),\end{aligned} \tag{5.10}$$

where λ_1, λ_2, λ_3, λ_4, and λ_5 are the trade-off parameters. When we calculate the third and fourth loss terms that take the minimum distance over the entire training dataset, the gradient computation would grow linearly with the size of the training set. However, this would be impractical during training for a large dataset. To solve this challenge, we propose relaxing the minimization to be over only the random minibatch used by the adopted gradient descent algorithm.

Discussion. In Eqn. (5.10), the value of M denotes the number of the considered concepts and determines the network structure of the model. If we directly follow existing concept-based explanation works [87, 116, 117] to manually define M, it will require significant human effort

and human involvement. Thus, in order to free human from tedious manual finding of a particular set of concepts (i.e., the value of M) in explaining the model's prediction behavior, we can follow the firefly neural architecture descent framework proposed in [118] to automatically determine the value of M (i.e., the set of satisfactory prototype-based concepts) via the automatic construction of the self-explanatory network architecture based on the desired properties. In addition, we can also follow the above proposed robust interpretability regularizer to provide certified robustness guarantees for the generated similarity results (i.e., $c(\boldsymbol{x}) = [c_1, \cdots, c_M]^T$).

5.3 Experiments

In this section, we conduct experiments to verify the effectiveness of the proposed method. Here we adopt three image datasets: the **MNIST** [22], **CIFAR-10** [119], and **AT&T**[2] datasets. The statistic information of the adopted datasets is given in Table 5.1.

Table 5.1: The statistic information of the adopted datasets.

	MNIST	CIFAR-10	AT&T
Dimension	$28 \times 28 \times 1$	$32 \times 32 \times 3$	$92 \times 112 \times 1$
Size	70,000	60,000	400
Classes	10	10	40
#Training	55,000	45,000	250
#Validation	5,000	5,000	50
#Testing	10,000	10,000	100

5.3.1 Visualization

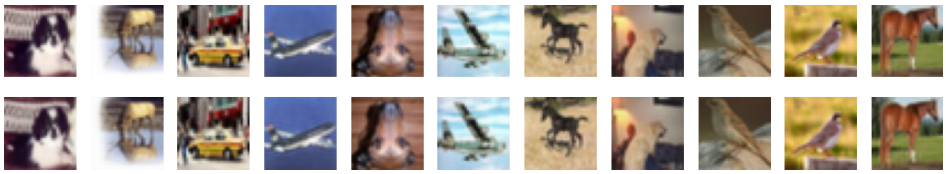
Firstly, we evaluate the performance of the trained autoencoder on the adopted datasets. The derived experimental results are reported in Figure 5.2. Take Figure 5.2a as an example, where the first line of the images is original images and the second line is the corresponding reconstructed images. From the reported experimental results in this figure, we can see that the reconstructed images are perceptually similar to the original images. We also report the derived reconstruction error and classification accuracy in Table 5.2. For example, in this table, the testing accuracy of the trained model on the MNIST dataset is 0.9816 and the autoencoder network achieves

[2] https://git-disl.github.io/GTDLBench/datasets/att_face_dataset/

	MNIST	CIFAR-10	AT&T
Restruction error	2.5851	3.2088	2.2170
Training Accuracy	0.9887	0.7899	0.8907
Validation Accuracy	0.9924	0.7926	0.9053 Spe-ML
Testing Accuracy	0.9816	0.7918	0.8978

Table 5.2: The reconstruction error and classification accuracy on the adopted datasets.

(a) MNIST



(b) CIFAR-10

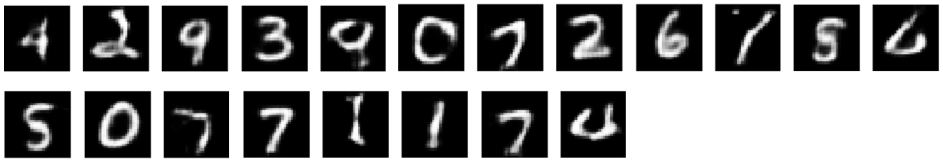
(c) AT&T

Figure 5.2: Reconstructed images on the adopted datasets.

a reconstruction error of 2.5851, which demonstrates that the proposed AutoRMI can achieve comparable accuracy performance to existing classification methods. Importantly, the good performance of the autoencoder component allows us to interpret the learned prototype-based concepts during model training.

Next, we visualize the learned prototype-based concepts. The obtained experimental results on the MNIST dataset are shown in Figure 5.3. In Figure 5.3a, we visualize the learned prototype-based concepts (learned in-process during model training) when $\lambda_2 = \lambda_3 = \lambda_6 = 0.05$. Note that these prototype-based concepts are decoded via the decoder. In Figure 5.3a, we can observe that the prototype-based concepts resemble real-world handwritten digits and give a high-level overview of the original data, due to the designed interpretability regularization term

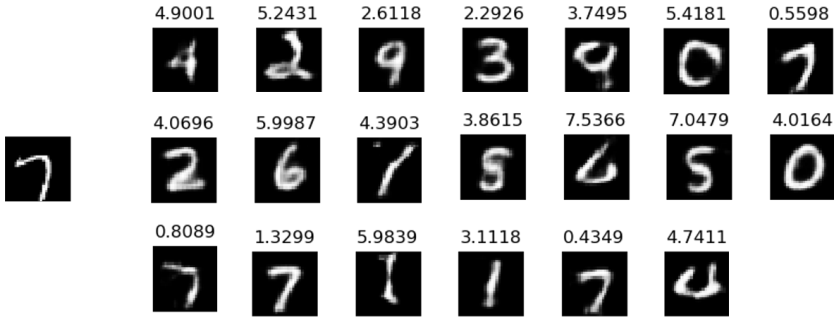
(i.e., $\mathcal{L}_2$ in Eqn. (5.2)). For comparison, we in Figure 5.3b also visualize the learned prototype-based concepts when we set $\lambda_2 = \lambda_3 = \lambda_6 = 0$ to remove the interpretability regularization term (i.e., $\mathcal{L}_2$ in Eqn. (5.2)). From the reported experimental results, we can see that when the interpretability regularizer are removed, the decoded concepts do not look like real-world images, which verifies that the proposed interpretability regularizer can guide the model to learn representative patterns during the training process.



(a) $\lambda_2 = \lambda_3 = \lambda_6 = 0.05$

(b) $\lambda_2 = \lambda_3 = \lambda_6 = 0$

Figure 5.3: The learned prototype-based concepts on the MNIST dataset where $\lambda_2 = \lambda_3 = \lambda_6 = 0.05$ and $\lambda_2 = \lambda_3 = \lambda_6 = 0$.



(a)

6.25e-7	6.57e-6	1.04e-7	2.38e-7	2.05e-7	8.94e-8	3.57e-6
2.04e-7	3.75e-7	2.98e-6	1.78e-7	8.90e-7	3.75e-7	5.13e-7
9.23e-6	1.19e-5	1.92e-5	6.55e-7	**0.999**	5.86e-7	-

(b)

Figure 5.4: Visualization results for the prediction result.

Then, we discuss how to use the learned prototype-based concepts to explain each predicted

classification result. In Figure 5.4, we present the visualization results for explaining the predicted classification result for a specific testing image of digit 7, which is shown on the left of Figure 5.4a. In Figure 5.4a, we give the distances (computed by the similarity layer) between the encoded representation of this testing image and each of the learned prototype-based concepts, and these distance values are shown above the decoded prototype-based concepts. From Figure 5.4a, we can see that the three prototype-based concepts that mostly resemble the testing image of digit 7 after decoding have the most shortest distances (i.e., 0.8089, 0.5598, and 0.4349). Importantly, the testing image of digit 7 is more closer to the third "7" concept (in the third line in Figure 5.4a) than the other two prototype-based concepts. From Figure 5.4b, we can also observe that compared with other concepts, this most closer concept also has the largest importance score (i.e., 0.999), which verifies that the proposed AutoRMI can capture the subtle differences within the same class. For this specific testing image, its prediction probability of class 7 is 99.98%.

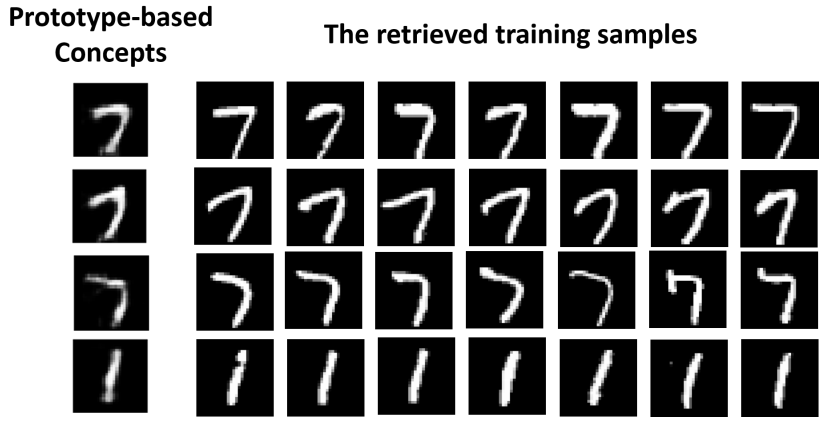


Figure 5.5: The retrieved training samples having the smallest distances from the learned concepts on the MNIST dataset.

Finally, for each learned prototype-based concept $\boldsymbol{p}_m$, we want to retrieve a subset of training samples that have the shortest distances from this prototype-based concept. Specifically, we aim to represent each learned prototype-based concept $\boldsymbol{p}_m$ by finding a subset of the input training dataset $\boldsymbol{X} = \arg\min_{\hat{\boldsymbol{X}} \subset \mathcal{X}, |\hat{\boldsymbol{X}}|=k} \sum_{\boldsymbol{x} \in \hat{\boldsymbol{X}}} c_m$, where k is pre-defined. Note that the the smaller the value of c_m, the more similar the prototype-based concept $\boldsymbol{p}_m$ and the retrieved training sample, the better the retrieved sample can represent this learned prototype-based concept $\boldsymbol{p}_m$. In this experiment, we set $k = 7$ and select the top-7 closest training examples for each prototype-based concept. The obtained experimental results on the adopted MNIST dataset are reported in Figure 5.5. The reported experimental results in this figure show that the learned

prototype-based concepts are representative examples. Additionally, from the reported experimental results in this figure, we can also see that the extracted prototype-based concepts and their corresponding retrieved close trained samples are visually similar, which means that they have the same patterns.

5.3.2 Robustness

Evaluation metric. Here, we consider one natural metric for quantifying the similarity between interpretations for two different samples, i.e., the top-k intersection. Specifically, we aim to see how many of the top-k concepts are no longer the top-k concepts after the adversarial perturbations. In many real-world settings, only the most important concepts are of explanatory interest. The lower the concepts in the intersection, the better the performance of the proposed method. In such settings, the adversarial attacker can launch the top-k attacks [69, 102, 112, 113, 114, 120, 121, 122, 123, 124] to decrease the relative importance of the k initially most important concepts. Hence, we propose to compute the size of intersection of the k most important concepts before and after the adversarial perturbations.

	MNIST		AT&T	
Methods	$\epsilon = 0.5$	$\epsilon = 1.0$	$\epsilon = 0.5$	$\epsilon = 1.0$
AutoRMI	0.89±0.08	0.85±0.10	0.73±0.08	0.67±0.05
Standard	0.79±0.13	0.70±0.06	0.69±0.09	0.62±0.08

Table 5.3: The change of explanations under different perturbation values.

Performance. To test the empirical robustness of the generated explanations against adversarial perturbations, we here used an ℓ_∞ attack. Here, the value of ϵ is varied from 0.5 to 1.0. Since there is no existing certified robustness work, in this experiment, we adopt the standard baseline (denoted as Standard), where we remove the robustness regularization term (i.e., $\lambda_5 = 0$). In Table 5.3, we report the obtained experimental results. From the reported experimental results in this table, we can observe that models trained with the proposed certified robust interpretability regularizer in Eqn. (5.9) perform better than the model obtained with standard training procedure, while the standard model (trained without the certified robust interpretability regularization term) is more vulnerable to adversarial perturbations. Additionally, we can also see that as the severity of adversarial perturbations (the value of ϵ) increases, the networks trained with the proposed method show significant performance improvement over the model

trained with standard training process.

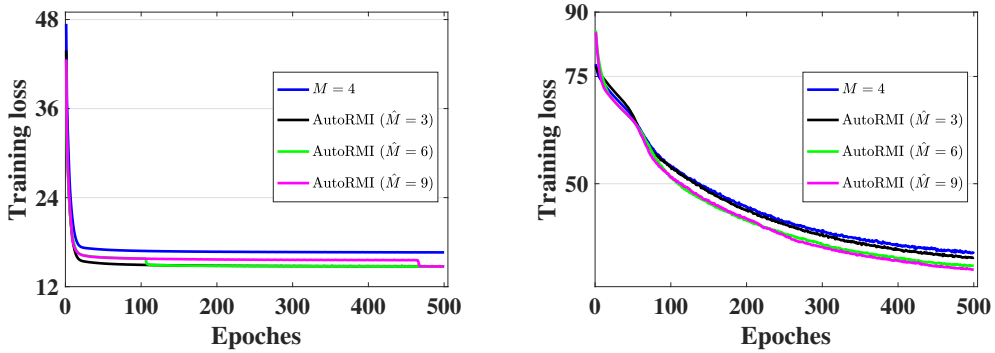
5.3.3 Architecture Search

Here, we evaluate the effectiveness of the proposed AutoRMI on the search of the network architectures. We also adopt the Wine Quality and Diabetic Retinopathy datasets [125]. We start with a small initial network with $M = 4$ and gradually increase the model size. Note that the value of $\hat{M}$ is the number of new neurons and hence determines the network structure (e.g., the concept and similarity layers). In Figure 5.6, we report the training loss of the proposed AutoRMI under different numbers of candidate grown neurons (i.e., different values of $\hat{M}$). Here, the value of $\hat{M}$ is varied from 3 to 9 for the adopted datasets. From this figure, we can see that the objective value gradually converges to 0 when increasing the training epochs, which also verifies that the convergence of AutoRMI can be guaranteed. In addition, we can also observe that the performance improves by even adding three new neurons. Furthermore, the reported experimental results demonstrate that the models trained with the selected network architectures perform better than that trained only with the initial values of M (i.e., $M = 4$ and $\hat{M} = 0$).

5.4 Related Work

Concept-based explanations have drawn much attention recently [86, 85, 96, 97, 98, 87, 95, 99]. Although these concept-based explanation methods are promising, their scalability is limited by the need for "humans-in-the-loop". The methods proposed in [85, 86, 96, 99] need human to manually define/extract concepts and quantify the importance score of each pre-defined concept in a post-hoc way. [97] directly performs the intervention of adding or removing a concept. [98] explains model decisions in terms of the importance of user-defined concepts. [95, 108] need human to be involved to manually pre-define the number of concepts. By assuming the existence of the concept representation, [87, 116, 117] manually define concepts and then use an intermediate set of human-specified concepts to predict the output task label. However, all of the above mentioned works heavily rely on experienced human experts who are expensive and hard to find. Furthermore, the above mentioned works also fail to address the certified robustness guarantees of the generated model explanations.

Recently, there have been a few efforts [104, 126, 100, 105, 106, 107] that have explored the robustness of model explanations. The authors in [104] propose a robust post-hoc feature-level



(a) MNIST (b) AT&T

(c) Diabetic Retinopathy (d) Wine Quality

Figure 5.6: The training loss of the proposed method under values of $\hat{M}$ on the adopted datasets.

explanation framework for constructing a global explanation. [126, 100, 105, 107] focus on the post-hoc gradient-based interpretation methods (e.g., Saliency Map) that are popular methods for deep learning interpretation. [106] interprets intermediate layers and defines robustness as the ability of an intermediate layer to be consistent in its recall rate for different random samples. However, their problem settings are significantly different from that of this work, and hence they cannot be directly applied here. In contrast, in this work, we directly guide models to automatically provide the concept-based explanations for each predicted decision, while guaranteeing the robustness of the generated explanations. Although [108] considers the intrinsic interpretation model, it only addresses the stability of the relevance scores and fails to provide robustness guarantees for the obtained interpretable representation. Furthermore, the robustness improvement of all of the aforementioned works cannot be certified – no provable guarantees can be given to verify their robustness. In fact, in practice, these uncertified methods may become vulnerable under stronger adversarial attacks.

5.5 Conclusions

In this chapter, we designed a novel automatic and robust self-explanatory method (AutoRMI) that can not only automatically provide the concept-based explanations without human interventions but also provide certified robustness guarantees for the generated concept-based explanations. Specifically, to free human from the tedious manual defining procedure, we first proposed a novel interpretability regularizer that guides the model to automatically extract the prototype-based concepts from the training data, which provide insights into representative patterns that are utilized by the model for classification. In addition, to promote certified robust interpretability, we also proposed a novel interval bound propagation based regularizer, which minimizes an upper bound on the maximum difference between any pair of explanation results when the input can be adversarially perturbed to provide verifiable robustness guarantees for the generated explanations. We also conducted experiments to demonstrate the effectiveness of the proposed method on real-world datasets and the experimental results show that our method can consistently achieve good performance.

Part III

Privacy-preserving Sharing of the Sensitive Information

Chapter 6

Pairwise Learning with Differential Privacy Guarantees

6.1 Introduction

As an important family of learning problems, *pairwise learning* has drawn much attention recently. Since pairwise learning involves a loss function depending on pairs of samples, it shows great advantage in modeling the relative relationship between pairs of samples over traditional pointwise learning (e.g., classification), in which the loss function only takes individual samples as the input. In practice, many learning tasks can be categorized as pairwise learning problmes. For instance, metric learning [6, 55, 3, 127, 8, 7] aims to learn a distance metric from a given collection of pair of similar/dissimilar samples that preserves the distance relation among the data, which can be formulated as a pairwise learning problem. Apart from metric learning, many other learning tasks, such as AUC maximization [128, 5] and ranking [9], can also be categorized as pairwise learning.

Existing pairwise learning algorithms can be roughly divided into two categories: *online* and *offline*. The online pairwise learning algorithms process the input data records in a sequential manner and iteratively update the model upon the arrival of each sample [128, 129]. In contrast, the offline pairwise learning algorithms require the entire training dataset ready before the learning process starts and take it as whole to update the model [56, 55].

Although the importance of pairwise learning has been recognized in many real-world applications, existing pairwise learning algorithms fail to take into consideration an important issue in their designs, that is, the protection of sensitive information in the training set. The training

datasets for pairwise learning are often collected from individual users and thus may contain private personal information. The models learned by such algorithms can implicitly memorize some details of the sensitive information, which undesirably offers opportunity for malicious parties to compromise the users' privacy. Taking the patient similarity learning task as example, a hospital may want to train a universal patient similarity learning model from patients (crossing many hospitals) so as to obtain a better understanding of the diseases and diagnoses. Due to trust to the hospital, patients may be willing to provide necessary information for such a learning process. However, without a proper mechanism, the patients' privacy may be breached when the trained model by the hospital is provided to other parties (such as medical research institutes or drug makers). This is because these parties can infer patients' private information using various attack techniques, such as model inversion attack [130] and membership attack [131]. Thus, without a convincing privacy-preserving mechanism, the patients may not be willing to participate in such a learning task. Hence, a big challenge facing pairwise learning is how to learn a model privately such that sensitive information cannot be inferred from the learned model.

To the best of our knowledge, no existing work has addressed the above challenge. This motivates us to design, in this chapter, privacy-preserving pairwise learning methods which can not only keep the sensitive information private but also guarantee good generalization performance. Among existing privacy-preserving strategies, differential privacy (DP) [132], as a rigorous notion for data privacy, can provide very rigid privacy and utility guarantee. Although various DP methods exist for (online) pointwise learning, such as objective perturbation or DP-SGD [133, 134, 135, 136], they cannot be applied to pairwise learning algorithms directly. This is mainly because the training sample pairs in pairwise learning algorithms are not i.i.d. and the loss function depends on more than one data records. In the light of the above challenges, in this chapter, we propose efficient differentially private algorithms for the aforementioned two types of pairwise learning problems. Our contributions can be summarized as follows:

- Firstly, we consider the pairwise learning problem in the online setting, and propose an (ϵ, δ)-DP algorithm called online pairwise private GIGA-Strongly convex method (**OnPairStrC**). This algorithm achieves a regret upper bound of $\tilde{O}(\frac{\sqrt{d}\sqrt{n}}{\epsilon})$ when the losses are strongly convex, where d is the feature dimension and n is the data size. We then extend this algorithm to general convex losses by proposing an algorithm called online pairwise private GIGA-convex method (**OnPairC**), which has a regret upper bound of $\tilde{O}(\frac{\sqrt{d}n^{\frac{3}{4}}}{\epsilon})$.

- Secondly, we study the pairwise learning problem in the offline setting. We show that it is possible to achieve generalization errors of $\tilde{O}(\frac{\sqrt{d}}{\sqrt{n\epsilon}})$ and $\tilde{O}(\frac{\sqrt{d}}{\sqrt[4]{n\epsilon}})$ for strongly and general convex loss functions respectively by adopting the results in the online settings. We then improve these bounds by proposing an offline pairwise private GIGA-Strongly convex algorithm (**OffPairStrC**) and an offline pairwise private GIGA-convex algorithm (**OffPairC**) for the two types of loss functions. Particularly, in the case of general convex loss functions, our improved algorithm can achieve a generalization error of $\tilde{O}(\frac{\sqrt{d}}{\sqrt{n\epsilon}})$.

6.2 Related Work

As mentioned earlier, there is no existing result on pairwise learning under the differential privacy model. Thus, we only compare ours with those private pointwise learning methods. There is a long list of papers on differentially private pointwise learning in the last decade which attack the problem from different perspectives. For DP pointwise learning with convex loss functions, there are a lot of works on it, such as [133, 137, 134, 138, 136]. However, all of the above results focus only on pointwise loss functions and cannot be extended to pairwise loss functions.

Differentially private pointwise learning in the online setting has also been studied previously [135, 139]. The works that are most related to ours are probably [135] and [139], where the authors gave a general framework for online convex optimization under differential privacy. However, there are some significant differences from ours. Firstly, [135] and [139] consider only pointwise loss functions while we study pairwise loss functions. Thus, their methods cannot be directly extended to pairwise loss functions, making them incomparable with ours; secondly, due to the differences in the structure of two problems and the definitions of the regret, the analyses of the upper bounds and the DP guarantees are quite different (see Remark 1 for more details).

6.3 Preliminaries

We say that two datasets D, D' are neighbors if they differ by only one entry, which is denoted as $D \sim D'$.

Definition 1 (Differential Privacy [132])**.** *A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private (DP) if for all neighboring datasets D, D' and for all events S in the output space of $\mathcal{A}$, the following holds*

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

When $\delta = 0$, $\mathcal{A}$ is ϵ-differentially private.

In this chapter we focus on (ϵ, δ)-DP and use Gaussian mechanism [132] to guarantee (ϵ, δ)-DP.

Definition 2 (Gaussian Mechanism)**.** *Given any function $q : \mathcal{X}^n \to \mathbb{R}^d$, the Gaussian mechanism is defined as $\mathcal{M}_G(D, q, \epsilon) = q(D) + Y$, where Y is drawn from Gaussian Distribution $\mathcal{N}(0, \sigma^2 I_d)$ with $\sigma \geq \frac{\sqrt{2\ln(1.25/\delta)}\Delta_2(q)}{\epsilon}$. Here $\Delta_2(q)$ is the ℓ_2-sensitivity of the function q, i.e., $\Delta_2(q) = \sup_{D \sim D'} ||q(D) - q(D')||_2$. Gaussian mechanism preserves (ϵ, δ)-differential privacy.*

Additionally, we also use zero Concentrated Differential Privacy (zCDP) [140] and its composition property to guarantee (ϵ, δ)-DP. Compared to directly using the composition property of DP, it has many advantages (see [141, 142] for more details).

Definition 3. *A randomized mechanism $\mathcal{A}$ is ρ-zCDP if, for all neighboring dataset D, D' and all $\alpha \in (1, \infty)$,*

$$D_\alpha(\mathcal{A}(D)||\mathcal{A}(D')) \leq \rho\alpha,$$

where $D_\alpha(\cdot||\cdot)$ is the α-Rényi Divergence [1] .

Lemma 1. *[140] Suppose that two mechanisms satisfy ρ_1-zCDP and ρ_2-zCDP, respectively. Then, their composition is $(\rho_1 + \rho_2)$-zCDP.*

Lemma 2. *[140] For a Gaussian mechanism $q(D) + Y$ with $Y \sim \mathcal{N}(0, \sigma^2 I_d)$, it satisfies $(\frac{\Delta_2^2(q)}{2\sigma^2})$-zCDP.*

Lemma 3. *[140] If a mechanism is ρ-zCDP, then it is $(\rho + 2\sqrt{\rho \log \frac{1}{\delta}}, \delta)$-DP for any $\delta > 0$.*

6.3.1 Private Pairwise Learning

Different from the pointwise loss function $\ell : \mathcal{C} \times \mathcal{D} \mapsto \mathbb{R}$, a pairwise loss function is a function on pairs of data records, *i.e.,* $\ell : \mathcal{C} \times \mathcal{D} \times \mathcal{D} \mapsto \mathbb{R}$, where $\mathcal{D}$ is the data universe. Given a dataset $D = \{z_1, z_2, \cdots, z_n\} \subseteq \mathcal{D}^n$ and a loss function $\ell(\cdot; \cdot, \cdot)$, its empirical risk can be defined as:

$$L(w; D) = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i} \ell(w; z_i, z_j). \tag{6.1}$$

[1] *For two distributions P and Q on Ω and $\alpha \in (1, \infty)$, the α-Rényi Divergence between P, Q is defined as $D_\alpha(P||Q) = \frac{1}{\alpha - 1} \log \int_\Omega P(x)^\alpha Q(x)^{1-\alpha} dx$.*

When the inputs are drawn i.i.d from an unknown underlying distribution $\mathcal{P}$ on $\mathcal{D}$, the population risk is

$$L_{\mathcal{P}}(w) = \mathbb{E}_{z_i, z_j \sim \mathcal{P}}[\ell(w; z_i, z_j)]. \tag{6.2}$$

Similar to the definition of private pointwise learning [134], we can define private pairwise learning as follows.

Definition 4 (Private pairwise learning)**.** *Let $\mathcal{C} \subseteq \mathbb{R}^d$ be a convex, closed and bounded constraint set, $\mathcal{D}$ be a data universe, and $\ell : \mathcal{C} \times \mathcal{D} \times \mathcal{D} \mapsto \mathbb{R}$ be a pairwise loss function. Also, let $D = \{z_1 = (x_1, y_1), z_2 = (x_2, y_2), \cdots, z_n = (x_n, y_n)\} \subseteq \mathcal{D}^n$ be a dataset with data records $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and labels (responses) $\{y_i\}_{i=1}^n \subset [-1, 1]^n$. Private pairwise learning is to find a private estimator $w_{priv} \in \mathcal{C}$ so that the algorithm is (ϵ, δ) or ϵ differential private and the error is minimized, where the error for an estimator w can be measured by either the optimality gap $Err_D(w) = L(w; D) - \min_{w \in \mathcal{C}} L(w; D)$ or the generalization error $Err_{\mathcal{P}}(w) = L_{\mathcal{P}}(w) - \min_{w \in \mathcal{C}} L_{\mathcal{P}}(w)$.*

In this chapter, we will focus on a special class of pairwise loss functions [2] which contains the loss functions of metric learning, AUC maximization and bipartite ranking.

Assumption 1. *For the loss function, we here assume that it has the form of $\ell(w; z, z') = \phi(Y(y, y')h(w; x, x'))$, and ℓ is a G-Lipschitz and L-smooth convex function over w, where $Y(y, y') = y - y'$ or $Y(y, y') = yy'$. In the experimental part, we will let ϕ be the logistic function, i.e., $\phi(x) = \log(1 + e^{-x})$.*

Example 1: Metric Learning [56] The goal here is to learn a Mahalanobios metric $M_W^2(x, x') = (x - x')^T W(w - x')$ using loss function $\ell(W; z, z') = \phi(yy'(1 - M_W^2(x, x'))$, where $y, y' \in \{-1, +1\}$. The constraint set $\mathcal{C}$ is $\mathcal{C} = \{W : W \in \mathbb{S}^d, \|W\|_F \leq 1\}$.

Example 2: AUC Maximization [128], Bipartite Ranking [143] The goal here is to maximize the area under the ROC curve for a linear classification problem with the constraint of $\|w\|_2 \leq 1$. Here $h(w; x, x') = w^T(x - x')$ and $\ell(w; z, z') = \phi((y - y')h(w; x, x'))$, where $y, y' \in \{-1, +1\}$.

6.3.2 Online Private Pairwise Learning

Here we follow online pairwise learning in [129]. An online learning algorithm $\mathcal{A}$ is given sequential access to a stream of elements $z_1, z_2, z_3, \cdots, z_n$. At each time step $t = 2, 3, \cdots, n$,

[2] We note that all the (ϵ, δ)-DP algorithms in this chapter can be extended to general pairwise loss functions, although the upper bounds of the generalization errors may differ.

the algorithm selects a parameter $w_{t-1} \in \mathcal{C}$ upon which the data record z_t is revealed, and the algorithm incurs the following penalty

$$\hat{L}_t(w_{t-1}, D_t) = \frac{1}{t-1} \sum_{i=1}^{t-1} \ell(w_{t-1}; z_t, z_i), \tag{6.3}$$

where $D_t = \{z_1, \cdots, z_t\}$. Thus, the online algorithm $\mathcal{A}$ maps a data sequence $\{z_1, \cdots, z_n\}$ to a sequence of parameters $\{w_1, \cdots, w_{n-1}\}$. In the non-private case, the goal is to select $\{w_1, \cdots, w_{n-1}\}$ so as to minimize the **regret**, *i.e.,*

$$\mathcal{R}_{\mathcal{A}}(n, D) = \sum_{t=2}^{n} \hat{L}_t(w_{t-1}, D_t) - \min_{w \in \mathcal{C}} \sum_{t=2}^{n} \hat{L}_t(w, D_t). \tag{6.4}$$

Moreover, if all data are chosen i.i.d from the distribution $\mathcal{P}$, we also want to minimize the **generalized regret**, *i.e.,*

$$\mathcal{R}_{\mathcal{P},\mathcal{A}}(n) = \sum_{t=2}^{n} L_{\mathcal{P}}(w_{t-1}) - (n-1) \min_{w \in \mathcal{C}} L_{\mathcal{P}}(w). \tag{6.5}$$

If ℓ is convex, then from (6.5) we have parameter $\bar{w} = \frac{w_1 + \cdots + w_{n-1}}{n-1}$ satisfies the following generalization error:

$$L_{\mathcal{P}}(\bar{w}) - \min_{w \in \mathcal{C}} L_{\mathcal{P}}(w) \leq \frac{\mathcal{R}_{\mathcal{P},\mathcal{A}}(n)}{n-1}. \tag{6.6}$$

However, under the differential privacy model, we need to guarantee that the output sequence $\{w_1, \cdots, w_{n-1}\}$ is DP. Thus, private pairwise learning in the online setting can be defined as follows:

Definition 5 (Online private pairwise learning)**.** *Let $Z = \{z_1, z_2, \cdots, z_n\}$ be any sequence of data records in the data universe $\mathcal{D}$. Let the sequence of outputs by algorithm $\mathcal{A}$ be $\mathcal{A}(Z) = \{w_1, \cdots, w_{n-1}\}$. Then, $\mathcal{A}$ is (ϵ, δ)-differentially private if given any other data sequence Z' which differs in at most one entry with Z, for all events S, we have $Pr[\mathcal{A}(Z) \in S] \leq e^{\epsilon} Pr[\mathcal{A}(Z') \in S] + \delta$. The goal of online private pairwise learning is to select private outputs $\{w_1, \cdots, w_{n-1}\}$ that minimizes the (generalized) regret.*

From above discussions on (6.5) and (6.6), we know that if the generalized regret is low, the algorithm will have a good performance on generalization theoretically. From this view, the online setting is more general. Thus, in the chapter, we will first consider the online private pairwise learning and provide regrets for both strongly and general convex losses. After that, we will study the problem in the offline setting.

6.4 Online Private Pairwise Learning

We first consider the case that the loss function is strongly convex. After that, we will use the regularization perturbation strategy in [139] to extend the resulting algorithm to general convex loss functions.

Our algorithm is inspired by the stability of Generalized Infinitesimal Gradient Ascent (GIGA) [144, 135], which is a well-known online convex algorithm (see Remark 1 for discussions on the difference of our algorithm with the previous ones). The main steps are given in Algorithm 2.

Algorithm 2 Online Pairwise Private GIGA-Strongly Convex (OnPairStrC)

1: **Input:** Privacy parameters ϵ and δ, sequence of data record $\{z_1, z_2, \cdots, z_n\}$, constrained convex set $\mathcal{C} \subset \mathbb{R}^d$, and pairwise loss function $\ell(\cdot;\cdot,\cdot)$.
2: **Parameters:** ℓ is G-Lipschitz, L-smooth and α-strongly convex over w. Step time $T_1 = \max\{\lceil \frac{16L^2}{\alpha^2} \rceil, 7\}$.
3: Compute ρ which satisfies $\rho + 2\sqrt{\rho \log(\frac{1}{\delta})} = \epsilon$.
4: **for** $t = 1, \cdots, T_1$ **do**
5: Receive the data record z_t (incurs penalty $\hat{L}_t(w_{t-1}, D_t)$ when $t \geq 2$).
6: Randomly choose a parameter $w_t \in \mathcal{C}$.
7: **end for**
8: **for** $t = T_1 + 1, \cdots, n$ **do**
9: Receive the data record z_t (incurs penalty $\hat{L}_t(w_{t-1}, D_t)$).
10: Set step size $\eta_t = \frac{t-1}{t-2}\frac{2}{\alpha t}$
11: $w_t = \Pi_{\mathcal{C}}[w_{t-1} - \eta_t \nabla \hat{L}_t(w_{t-1}, D_t)]$, where $\Pi_{\mathcal{C}}$ is the projection onto the set $\mathcal{C}$.
12: Set $\sigma_t^2 = \frac{32G^2(n-T_1)}{\alpha^2 t^2 \rho}$. Let $\tilde{w}_t = w_t + n_t$, where $n_t \sim \mathcal{N}(0, \sigma_t^2 I_d)$.
13: Output $w_t = \arg\min_{w \in \mathcal{C}} \|w - \tilde{w}_t\|_2^2$.
14: **end for**

We call the above algorithm excluding the portion of random perturbation (*i.e.,* steps 12 and 13) **Pairwise GIGA**. The following lemma gives an upper bound on the ℓ_2-norm sensitivity of the output in the t-th iteration of Pairwise GIGA, which is to ensure (ϵ, δ)-DP of Algorithm 2.

Lemma 4. *Let $\mathcal{A}_t(D_t)$ denote the output of Pairwise GIGA in the t-th iteration. Then, under the assumption of Algorithm 2, for any $t \geq 1$ and $D_t \sim D_t'$,*

$$\|\mathcal{A}_t(D_t) - \mathcal{A}_t(D_t')\|_2 \leq \frac{8G}{\alpha t}.$$

In Algorithm 2, steps 4 to 7 seem weird due to the random sampling of w_{t-1}. However, as we see from the proof of Lemma 4, this condition is necessary for the stability analysis. Moreover, the similar steps have also been adopted in some algorithms on DP online learning, such as [139, 135].

Theorem 6. *Under Assumption 1 and the assumption that the loss function ℓ is α-strongly convex, for any $0 < \epsilon, \delta \leq 1$, Algorithm 2 is (ϵ, δ)-differentially private.*

Note that to guarantee DP, we first transfer (ϵ, δ)-DP to ρ-zCDP by Lemma 3, and then use composition theorem to make Algorithm 2 be ρ-zCDP (*i.e.,* we make each iteration $T_1 + 1 \leq t \leq n$ be $\frac{\rho}{n-T_1}$-zCDP). It is easy to see that in this case the variance of the noise satisfies $\sigma_t^2 = \frac{32G^2(n-T_1)}{\alpha^2 t^2(\sqrt{\log(1/\delta)+\epsilon}-\sqrt{\log(1/\delta)})^2}$. When $\frac{\epsilon}{\log(1/\delta)} \ll 1$ (this case will always holds since in practice we select $\epsilon = 0.1 \sim 5$ and $\delta = \frac{1}{n}$), by Taylor expansion of $\sqrt{1+x}$, we have $(\sqrt{\log(1/\delta)+\epsilon} - \sqrt{\log(1/\delta)})^2 \simeq \frac{\epsilon^2}{4\log(1/\delta)}$. Thus in total, we have $\sigma_t^2 \simeq \frac{128G^2(n-T_1)\log(1/\delta)}{\alpha^2 t^2 \epsilon^2}$.

The following theorem shows an upper bound on the regret of Algorithm 2, which can be transformed to generalized error (we will show it in the following section).

Theorem 7. *Under the assumptions in Theorem 6 and the additional condition of $\frac{\epsilon}{\log \frac{1}{\delta}} \ll 1$, Algorithm 2 has the following upper bound on the regret of its outputs*

$$\mathcal{R}_{\mathcal{A}}(n, D) \quad \leq \quad O\left(\frac{G^2\sqrt{d}\log^{1.5}\frac{n}{\zeta}\sqrt{n}\sqrt{\log\frac{1}{\delta}}}{\alpha\epsilon} \quad + \quad \frac{GL^2}{\alpha^2}\|\mathcal{C}\|_2 \quad + \quad \frac{G^2\log n}{\alpha}\right) \quad (6.7)$$

with probability at least $1 - \zeta$.

Remark 1. *We note that [135] used the differentially private version of GIGA and IGD [145] in their DP pointwise learning. But their Private GIGA or IGD algorithm is quite different from our method of OnPairStrC (Algorithm 2). Firstly, [135] needs to assume that each loss function $\hat{L}_t$ is independent (see the proofs of Lemma 4 and Lemma 5 in [135]), which means that it is only applicable to pointwise loss functions. However, in our problem, the penalty function (6.3) depends on previous data records, which means that it is much more complicated than the case in [135]. Thus, we need a much finer and more different analysis on the stability of Pairwise GIGA. Also, the parameters of the step size η_t and time step T_1 are quite different from those in [135]. Additionally, in order to show the power of our method, we also consider the case with additional finite buffer constraint, which has not been studied in [135]. Thus, our method is more general.*

Secondly, the upper bound (6.7) on the regret of our algorithm is less than that in [135] with a factor of $\log \frac{n}{\delta}$*. This is due to the fact that we use the composition property of zCDP instead of advanced composition theorem of DP [146].*

*Thirdly, since the definition of regret is different from that in pointwise learning [135], the same upper bound (*i.e., $\tilde{O}(\frac{\sqrt{dn}}{\epsilon})$*) on the regret for strongly convex losses are actually incomparable.*

We now use the perturbation strategy in [139] to obtain result for general convex losses.

Algorithm 3 Online Pairwise Private GIGA-Convex (OnPairC)

1: **Input:** Privacy parameters ϵ and δ, sequence of data records $\{z_1, z_2, \cdots, z_n\}$, constrained convex set $\mathcal{C}$, pairwise loss $\ell(\cdot;\cdot,\cdot)$, and a parameter α to be defined later.
2: **Parameters:** ℓ is G-Lipschitz, L-smooth and convex over w.
3: Randomly select a point $w_0 \in \mathcal{C}$. Let $\tilde{\ell}(w; z, z') = \ell(w; z, z') + \frac{\alpha}{2}\|w - w_0\|_2^2$.
4: Run Algorithm 2 with loss $\tilde{\ell}$, which is $\tilde{G} = G + \alpha\|\mathcal{C}\|_2$-Lipschitz, $\tilde{L} = L + \alpha$-smooth and α-strongly convex.

Theorem 8. *Let* ℓ *be a pairwise loss function satisfying Assumption 1. Then, for any* $0 < \epsilon, \delta \leq 1$*, Algorithm 3 is* (ϵ, δ)*-DP. Moreover, if* $\frac{\epsilon}{\log \frac{1}{\delta}} \ll 1$ *and take* $\alpha = O(\frac{1}{\sqrt[4]{n}})$*, then with probability at least* $1 - \zeta$*, the following upper bound on regret for the outputs holds:*

$$\mathcal{R}_{\mathcal{A}}(n, D) \leq O(\frac{L^2 G^2 \|\mathcal{C}\|_2^2 \sqrt{d} \log^{1.5} \frac{n}{\zeta} n^{\frac{3}{4}} \sqrt{\log \frac{1}{\delta}}}{\epsilon}). \tag{6.8}$$

Comparing (6.8) with (6.7), we can see that for strongly convex pairwise loss functions, the average regret, *i.e.,* $\frac{\mathcal{R}_{\mathcal{A}}(n)}{n-1}$, is upper bounded by $\tilde{O}(\frac{\sqrt{d}}{\sqrt{n}\epsilon})$, while for general convex ones, it is $\tilde{O}(\frac{\sqrt{d}}{\sqrt[4]{n}\epsilon})$. This is the same as in the case of pointwise loss functions [139].

6.5 Offline Private Pairwise Learning

6.5.1 Generalization Error Induced by Generalized Regret

We first observe that Algorithm 2 and 3 preserve (ϵ, δ)-DP in the offline settings. Also, as discussed in (6.5) and (6.6), if we get the generalized regret for the output $\{w_1, w_2, \cdots, w_{n-1}\}$, we can easily obtain a generalization error by (6.6). By a theorem in [129], we can have the following generalization bounds for $\bar{w} = \frac{w_1 + \cdots + w_{n-1}}{n-1}$ of Algorithm 2 and 3. Before this, we

first let the Rademacher averages of the pairwise loss functions class $\ell \circ \mathcal{C} := \{(z, z') \mapsto \ell(w; z, z'), w \in \mathcal{C}\}$ be denoted by the following [129]:

$$\mathcal{R}_n(\ell \circ \mathcal{C}) = \mathbb{E}[\sup_{w \in \mathcal{C}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i \ell(w; z, z_i)], \tag{6.9}$$

where $\{\epsilon_i\}_{i=1}^n$ are the Rademacher variables, and the expectation is over $\{\epsilon_i\}_{i=1}^n, z, \{z_i\}_{i=1}^n$.

Theorem 9. *Under Assumption 1, the parameter $\bar{w} = \frac{w_1 + \cdots + w_{n-1}}{n-1}$ satisfies the following generalization error for loss function ℓ with probability at least $1 - 2\zeta$ if $\frac{\epsilon}{\log \frac{1}{\delta}} \ll 1$, where $w_1, w_2, \cdots, w_{n-1}$ are the outputs of Algorithm 3 (Algorithm 2 for strongly convex loss functions),*

$$Err_{\mathcal{P}}(\bar{w}) \quad \leq \quad O\big(\frac{\sum_{t=2}^{n} \mathcal{R}_{t-1}(\ell \circ \mathcal{C})}{n-1} \quad + \quad \frac{L^2 G^2 \|\mathcal{C}\|_2^2 \sqrt{d} \log^{1.5} \frac{n}{\zeta} \sqrt{\log \frac{1}{\delta}}}{\epsilon \sqrt[4]{n}}\big). \tag{6.10}$$

Moreover, if the loss is α-strongly convex, then we have:

$$Err_{\mathcal{P}}(\bar{w}) \leq O\big(\frac{1}{n-1} \sum_{t=2}^{n} \mathcal{R}_{t-1}(\ell \circ \mathcal{C}) + \frac{G^2 L^2 \|\mathcal{C}\|_2 \sqrt{d} \log^{1.5} \frac{n}{\zeta} \sqrt{\log \frac{1}{\delta}}}{\alpha^2 \epsilon \sqrt{n}}\big). \tag{6.11}$$

Remark 2. *Note that there are many problems whose Rademacher average is $\mathcal{R}_n(\ell \circ \mathcal{C}) = O(\frac{\sqrt{d}}{\sqrt{n}})$, e.g. Example 1 and 2 [129]. Thus for Example 1, the generalization error is $\tilde{O}(\frac{d}{\epsilon \sqrt[4]{n}})$ for logistic loss while it is $\tilde{O}(\frac{d}{\epsilon \sqrt{n}})$ if adding an additional Frobenious regularization to the losses. Similar result holds for Example 2, where the generalization error is $\tilde{O}(\frac{\sqrt{d}}{\epsilon \sqrt[4]{n}})$ while it is $\tilde{O}(\frac{\sqrt{d}}{\epsilon \sqrt{n}})$ in the case with additional ℓ_2-norm regularization.*

6.5.2 Improved Upper Bounds by Offline Differentially Private Algorithms

Inspired by the sensitivity of Pairwise GIGA in Lemma 4 and Theorem 9, we propose an offline DP algorithm which has better upper bounds compared to (6.10) and (6.11). The basic idea is to use output perturbation. Specifically, we first run Pairwise GIGA in the offline settings and then add some Gaussian noises to $\tilde{w} = \frac{w_1 + \cdots + w_n}{n}$ to keep the algorithm (ϵ, δ)-DP, since the sensitivity of $\tilde{w}$ is based on each w_i, which can be obtained by Lemma 4. For general convex loss functions, we can still use the perturbation idea, which is the same as in Algorithm 3. See Algorithm 4 and 5 for details.

The reason that we can improve the generalization error is due to the following fact. From Algorithms 2 and 3, we can see that the output sequences $\{w_1, w_2, \cdots, w_{n-1}\}$ satisfy the conditions of (ϵ, δ)-DP in each iteration. However, in the offline setting, we only need to ensure

that the final output is DP. Thus, instead of adding noise in each iteration, we can add noises only once to the final output, meaning that we can add a smaller scale of noises compared to the online ones.

Theorem 10. *For any $0 < \epsilon, \delta \leq 1$, Algorithm 4 is (ϵ, δ)-DP for any α-strongly convex loss functions satisfying Assumption 1. Moreover, if $\frac{\epsilon}{\log \frac{1}{\delta}} \ll 1$, then with probability at least $1 - 2\zeta$, the output $\hat{w}$ satisfies:*

$$Err_{\mathcal{P}}(\hat{w}) \leq O\Big(\frac{\sqrt{d}G^2\|\mathcal{C}\|_2 \log \frac{n}{\zeta}\sqrt{\log \frac{1}{\delta}}}{\alpha n \epsilon} + \frac{1}{n}\sum_{t=1}^{n}\mathcal{R}_t(\ell \circ \mathcal{C})\Big). \tag{6.12}$$

Algorithm 5 is (ϵ, δ)-DP for any convex loss functions satisfying Assumption 1 if $\alpha = O(\frac{1}{\sqrt{n}})$. Moreover, if $\frac{\epsilon}{\log \frac{1}{\delta}} \ll 1$, then with probability at least $1 - 2\zeta$, the output $\hat{w}$ satisfies:

$$Err_{\mathcal{P}}(\hat{w}) \quad \leq \quad O\Big(\frac{\sqrt{d}G^2\|\mathcal{C}\|_2^2 \log \frac{n}{\zeta}\sqrt{\log \frac{1}{\delta}}\log n}{\sqrt{n}\epsilon} \quad + \quad \frac{1}{n}\sum_{t=1}^{n}\mathcal{R}_t(\ell \quad \circ \quad \mathcal{C})\Big). \tag{6.13}$$

From Theorem 10, we can see that for strongly and general convex loss functions, the bounds in (6.13) and (6.12) are respectively lower than those in (6.10) and (6.11). Specifically, for general convex loss functions, we can improve the upper bound from $\tilde{O}(\frac{\sqrt{d}}{\epsilon\sqrt[4]{n}})$ to $\tilde{O}(\frac{\sqrt{d}}{\epsilon\sqrt{n}})$ if $\mathcal{R}_n(\ell \circ \mathcal{C}) = O(\frac{\sqrt{d}}{\sqrt{n}})$.

Algorithm 4 Offline Pairwise Private GIGA-Strongly Convex (OffPairStrC)

1: **Input:** Privacy parameters ϵ and δ, sequence of data $\{z_1, z_2, \cdots, z_n\}$, constrained convex set $\mathcal{C}$, pairwise loss $\ell(\cdot;\cdot,\cdot)$, and step number $T_1 = \max\{\lceil \frac{16L^2}{\alpha^2} \rceil, 7\}$.
2: **Parameters:** ℓ is G-Lipschitz, L-smooth and α-strongly convex over w.
3: Randomly sample $w_1, \cdots, w_{T_1} \in \mathcal{C}$.
4: **for** $t = T_1 + 1, \cdots, n$ **do**
5: Set step size $\eta_t = \frac{t-1}{t-2}\frac{2}{\alpha t}$.
6: Update
$$w_t = \arg\min_{w \in \mathcal{C}} \|w - (w_{t-1} - \eta_t \nabla \hat{L}_t(w_{t-1}, D_t))\|_2^2.$$
7: **end for**
8: Let $\tilde{w} = \frac{w_1 + \cdots + w_n}{n}$.
9: Denote $\bar{w} = \tilde{w} + \sigma$, where $\sigma \sim \mathcal{N}(0, \frac{128G^2 \log^2 n \log(1.25/\delta)}{\alpha^2 n^2 \epsilon^2} I_d)$.
10: Return $\hat{w} = \arg\min_{w \in \mathcal{C}} \|w - \bar{w}\|_2^2$.

Algorithm 5 Pairwise Private GIGA-Convex (OffPairC)

1: **Input:** Privacy parameters ϵ and δ, sequence of data $\{z_1, \cdots, z_n\}$, constrained convex set $\mathcal{C}$, pairwise loss function $\ell(\cdot;\cdot,\cdot)$, and a parameter α to be defined later.
2: **Parameters:** ℓ is G-Lipschitz, L-smooth and convex over w.
3: Let $\tilde{\ell}(w; z, z') = \ell(w; z, z') + \frac{\alpha}{2}\|w - w_0\|_2^2$, w_0 is any point in $\mathcal{C}$.
4: Run Algorithm 4 with loss $\tilde{\ell}$, which is $\tilde{G} = G + \alpha\|\mathcal{C}\|_2$-Lipschitz, $\tilde{L} = L + \alpha$-smooth and α-strongly convex.

6.6 Experiments

In this section, we empirically evaluate the performance of the proposed differentially private algorithms on real-world datasets. We take two popular pairwise learning tasks, i.e., AUC maximization and metric learning, as examples. All of the experiments in this chapter are conducted over 20 runs of different random permutations for each adopted dataset, and we report the averaged results.

6.6.1 Experimental Setup

Datasets

We use four real-world datasets that are widely adopted in pairwise learning tasks. These datasets are the **Diabetes** dataset, the **Diabetic Retinopathy** dataset, the **Hepatitis** dataset and the **Cancer** dataset [147].

Performance Measures

To evaluate the performance of the proposed algorithms, we use the following measures:

1. *AUC*: For AUC maximization task, we report the AUC measurement [128] for each of the proposed algorithms over every adopted dataset. A larger AUC value means that the corresponding AUC maximization algorithm can generate more accurate results.

2. *Classification Accuracy*: For metric learning task, we calculate the classification accuracy that is defined as the percentage of the correctly classified samples in the test set. The less the classification accuracy, the worse the performance of the proposed algorithm. In this chapter, the KNN classifier is adopted to assign labels to the test samples. For the KNN classifier, we set K to be 3.

3. *Objective function value*: For both metric learning task and AUC maximization task, we also report the objective function value of the proposed differentially private algorithms. A smaller objective function value means that the original pairwise learning model is less perturbed.

Baselines

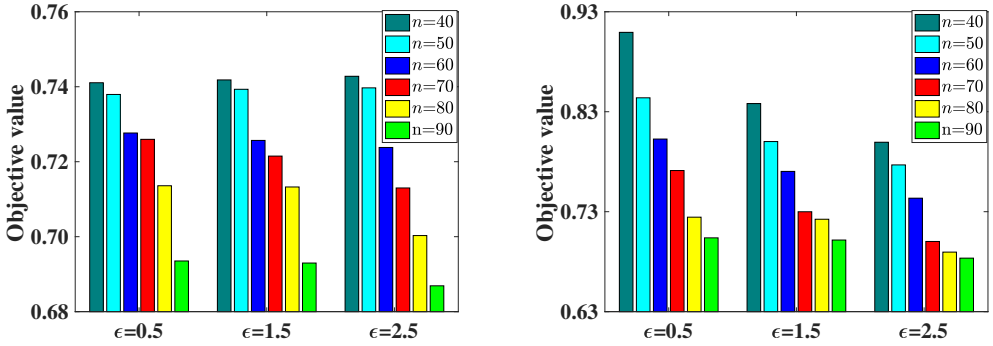
Since there is no existing work that addresses the privacy issue in pairwise learning, in experiments, we take the original pairwise learning algorithms that do not take any actions to protect the private information as the baselines. We denote the baseline methods as **NonPrivate**, which is the GIGA for pairwise loss functions [129].

6.6.2 Experiments for AUC Maximization

We first evaluate the performance of the proposed differentially private pairwise learning algorithms (i.e., OnPairStrC, OnPairC, OffPairStrC and OffPairC) for AUC maximization task (see Example 2 for the problem formulation). We add additional ℓ_2 regularization $\frac{\lambda}{2}\|w\|_2^2$ with $\lambda = 10^{-3}$ to loss function for the strongly convex case.

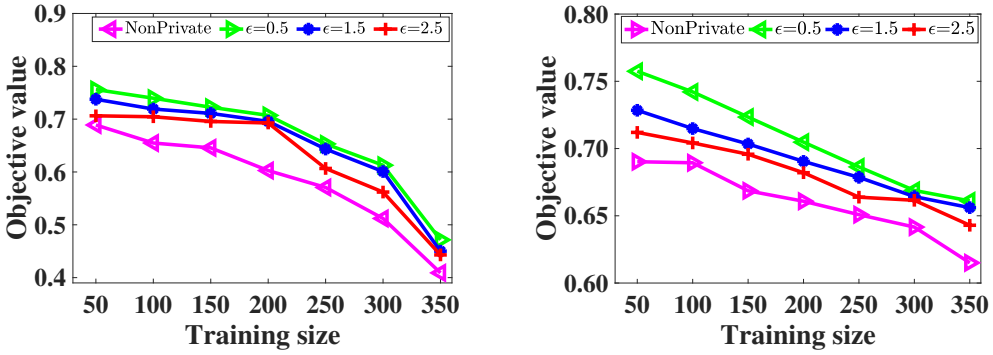
We study the effect of the training size n and the privacy parameter ϵ on the performance of the proposed OnPairStrC, OnPairC, OffPairStrC and OffPairC algorithms. Here we fix $\delta = \frac{1}{n}$ and consider three cases where the value of parameter ϵ is set to be 0.5, 1.5 and 2.5, respectively. For OnPairStrC and OffPairStrC, we vary the training size from 40 to 90 and conduct the experiment on the Hepatitis and Cancer datasets. For OnPairC and OffPairC, the experiment is conducted on the Diabetes and Diabetic Retinopathy datasets and we vary the training size from 50 to 350. In Figure 6.1 and Figure 6.2, we respectively report the objective values of OnPairStrC and OnPairC. The experimental results show that the larger the value of the training size n, the smaller the objective value. Additionally, when n is fixed, the smaller the value of ϵ, the larger the objective value is. The performance of the proposed algorithms are comparable with that of the baseline, which can be observed from Figure 6.2. The results for OffPairStrC and OffPairC are shown in Figure 6.3 and Figure 6.4, respectively. Figure 6.3 shows the objective value of OffPairStrC when the training size varies and Figure 6.4 reports the AUC measurement of OffPairC. The results in the two figures also show that the larger the training size is or privacy parameter ϵ is, the higher the AUC measurement value is, which means that the proposed algorithm is less perturbed and more accurate. These experimental results verify that the proposed online differential private algorithms can achieve good utility while guaranteeing strong privacy

protection when they are applied to AUC maximization task.



(a) Hepatitis (b) Cancer

Figure 6.1: The objective value of OnPairStrC for AUC maximization.

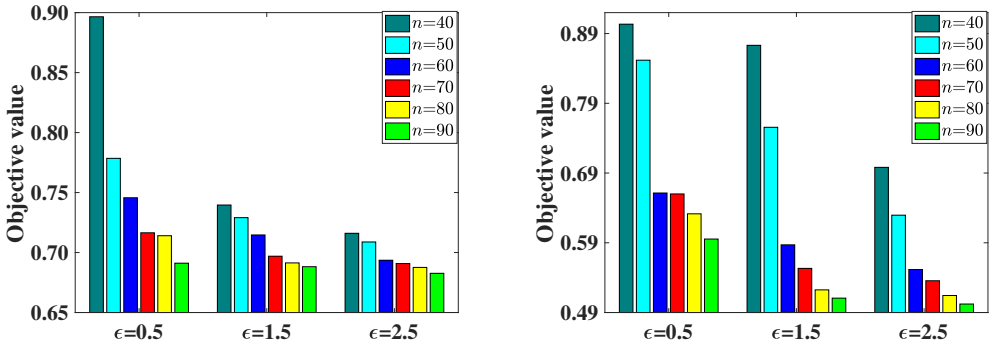


(a) Diabetes (b) Diabetic Retinopathy

Figure 6.2: The objective value of OnPairC for AUC maximization.

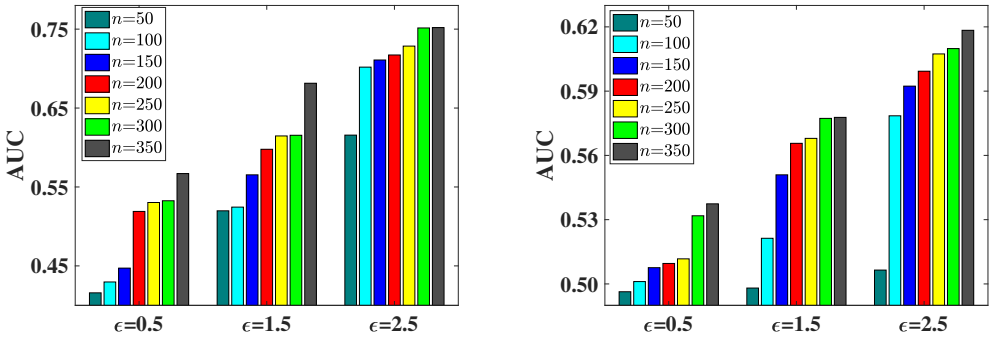
6.6.3 Experiments for Metric Learning

Next, we evaluate the performance of the proposed differentially private pairwise learning algorithms for the metric learning task (see Example 1 for the problem formulation). We add additional Frobenius norm $\frac{\lambda}{2}\|W\|_F^2$ to the loss function for the strongly convex case, where $\lambda = 10^{-3}$. Similar to the experiments for AUC maximization, we evaluate the effect of the privacy parameter ϵ and the training size n. Due to the space limit, in this section, we only report the experimental results for general convex pairwise learning algorithms, i.e., OnPairC and OffPairC.



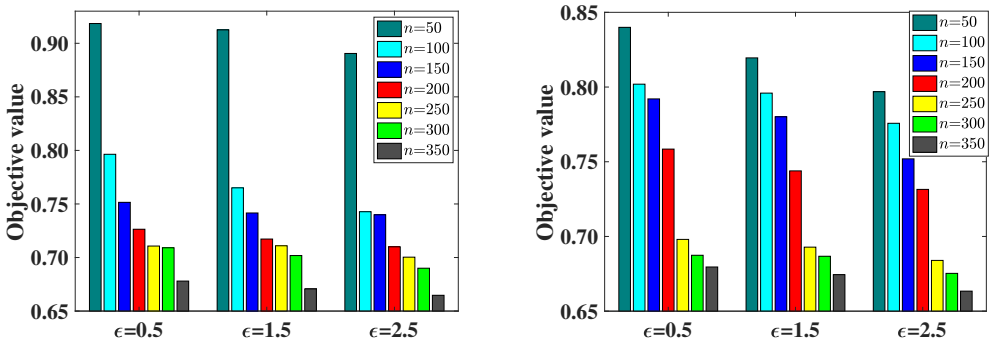
(a) Hepatitis (b) Cancer

Figure 6.3: The objective value of OffPairStrC for AUC maximization.



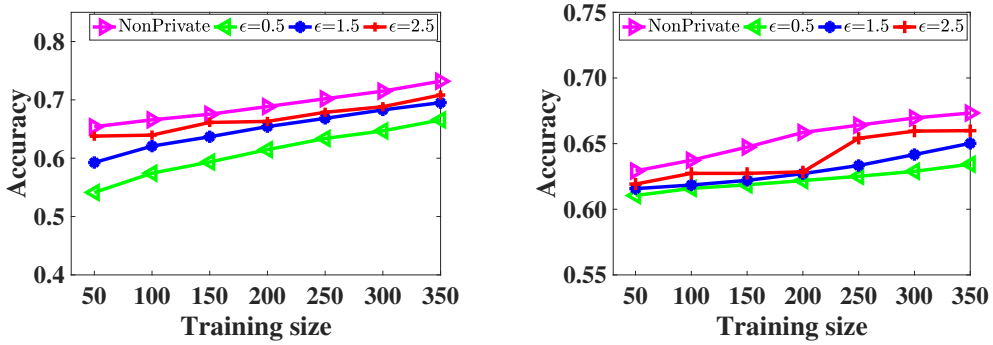
(a) Diabetes (b) Diabetic Retinopathy

Figure 6.4: The AUC measurement of OffPairC.



(a) Diabetes (b) Diabetic Retinopathy

Figure 6.5: The objective value of OnPairC for metric learning task under different training sizes.



(a) Diabetes (b) Diabetic Retinopathy

Figure 6.6: The classification accuracy of OffPairC for metric learning task under different training sizes.

In these experiments, the value of δ is fixed as $\frac{1}{n}$, and we consider three cases where the parameter ϵ is set to be 0.5, 1.5 and 2.5, respectively. We first calculate the objective value of OnPairC when the training size varies from 50 to 350, and the results on the Diabetes and Diabetic Retinopathy datasets are shown in Figure 6.5. As for the offline algorithm OffPairC, we report the classification accuracy in Figure 6.6. As we can see, the derived experimental results are similar to that for AUC maximization. The proposed algorithms perform competitively with the baseline when we vary the values of n and ϵ.

6.7 Conclusions

In this chapter, we consider the pairwise learning problems in both online and offline settings. For the online setting, we first propose an (ϵ, δ)-DP algorithm (called OnPairStrC) for the strongly convex loss functions, and then extend this algorithm to general convex loss functions by proposing another differentially private algorithm (called OnPairC). For the offline setting, we also propose two differentially private algorithms (called OffPairStrC and OffPairC) for strongly convex loss functions and general convex loss functions, respectively, and then give their regret upper bounds. The experimental results on real-world datasets not only confirm our theoretical analysis but also demonstrate the effectiveness of the proposed algorithms in real-world applications.

Chapter 7

Privacy-preserving Synthesizing for Crowdsourced Data

7.1 Introduction

In recent years, crowdsourcing has emerged as a popular and fast paradigm to solve many challenging data analysis tasks. Through the power of the crowd, the data collectors (e.g., hospitals, foundations and government agencies) can easily obtain large amounts of useful information. At the same time, the proliferation of new information techniques enables these data collectors to easily share their data that are collected from a crowd of users (e.g., patients, customers) with researchers or data analyzers. From such a wealth of shared data, researchers or data analyzers can discover useful knowledge or patterns to improve the quality of products, the management of public health, and so on. For example, in healthcare applications, the adverse events about a new drug can be easily collected by the hospitals from different patients. If the hospitals are willing to share these medical data, it would be very useful for the drug makers or medical research institutions to understand the efficacy of the drug.

Although the sharing of crowdsourced data brings many benefits, it may introduce privacy issues [148, 149, 150, 151]. Considering the above example, the hospital aims to collect the adverse events about a new drug from different patients. The patients usually trust the hospital and are willing to provide all the requested information. But if the hospital directly releases the patients' medical data to the drug makers, the private information of patients would be disclosed. Without effective privacy-preserving mechanisms, the patients may not allow their data to be released. Thus, it is essential to address how to enable the data collectors to release

the crowdsourced data without disclosing users' private information.

Among existing privacy-preserving techniques, differential privacy (DP) has drawn significant attention as it can provide very rigorous privacy and utility guarantee [152]. However, this technique has several practical limitations when it is applied in the setting of releasing crowdsourced data. First of all, since the crowdsourced data on an object (e.g., the new drug) are usually collected from multiple users or sources, there inevitably exist conflicts among these data. The reasons include incomplete views of observations, environment noise, different knowledge bases and even the intent to deceive, etc. Directly applying DP on these data can not eliminate the conflicts, and this will certainly degrade the accuracy of the data analysis results. Additionally, DP is usually achieved by adding noise following the Laplace or exponential mechanisms [152]. The noise scale introduced by the Laplace mechanism is proportional to the number of data records, and such noise may make the data useless considering that crowdsourced dataset usually contains large amounts of data records. Although the noise introduced by the exponential mechanism does not depend on the number of data records, it depends on the domains of the input data [153], which may also make the crowdsourced data useless because these data usually have large domains.

To address the above challenges, in this chapter, we propose a novel sampling-based **pri**vacy-aware **s**ynthesizing method for **crowd**sourced data (**PrisCrowd**). In this method, the data collector first learns the underlying patterns (i.e., densities) of the data for the objects through assigning each user a fine grained weight (or reliability degree) on each object. Then, for each object, the data collector samples a set of candidate synthetic data from the learned density. Finally, these synthetic data are subjected to our proposed privacy test, and the data collector only releases the synthetics that can pass the privacy test. The proposed method can not only extract high quality crowdsourced data via differentiating each user's fine grained reliability degrees on different objects but also achieve DP without injecting noise to the data. Both theoretical analysis and extensive experiments on real-world datasets are provided to verify the desirable performance of the proposed method.

7.2 Problem Setting

This chapter considers a data releasing scenario, where a crowd of users and a data collector are involved. The users (or data sources) are the individuals (e.g., patients, customers) who can observe some objects (e.g., drugs, commodities) and provide claims for them. The data collector is an individual or institution (e.g., a hospital, an online store) who can collect the claims for

these objects from a crowd of users and then release these claims to the public either voluntarily or for financial incentives. Here, we assume that the collector is trusted and the security threats mainly come from the public.

Problem formulation. Suppose there are N objects $\mathcal{O} = \{o_i\}_{i=1}^{N}$ which are observed by M users $\mathcal{U} = \{1, 2, ..., M\}$. For each object o_i, the claims of users are denoted as $\mathcal{X}_i = \{x_{i,s}\}_{s \in \mathcal{U}_i}$, where $x_{i,s}$ represents the claim provided by user s for object o_i and $\mathcal{U}_i$ represents the set of users who provide claims for this object. The claims collected by the data collector from all users are denoted as $\mathcal{X} = \{\mathcal{X}_i\}_{i=1}^{N}$, which need to be released to the public. Our goal in this chapter is to design a mechanism based on which the data collector can release users' claims with strong privacy protection for their private information, while at the same time, the data analyzer can achieve good utility from the released data.

7.3 Preliminary

Definition 6 (Differential Privacy [152])**.** *A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for all neighboring datasets $D, D' \in \mathcal{X}^n$ and for all events S in the output space of $\mathcal{A}$, the following holds:* $\Pr(\mathcal{A}(D) \in S) \leq e^{\epsilon} \Pr(\mathcal{A}(D') \in S) + \delta$.

The kernel density estimation (KDE) is a statistically-sound method to estimate a continuous distribution. Suppose there are n independent observations $X = \{x_1, ..., x_n\} \in \mathbb{R}^d$ following an unknown true density $f^*(x)$. The standard KDE $\tilde{f}(x)$ for the estimation of $f^*(x)$ at those points is defined as $\tilde{f}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{K}_{\mathcal{H}}(x, x_i)$. The following assumption will be used throughout the chapter.

Assumption 2. *For a vector $x_i \in \mathbb{R}^d$, we assume that the kernel function satisfies $\mathcal{K}_{\mathcal{H}}(x, x_i) = \mathcal{K}_{\mathcal{H}}(x - x_i)$. Furthermore, $\mathcal{K}_{\mathcal{H}}(x - x_i)$ is essentially a bump centered at x_i. More specifically, we take $\mathcal{K}_{\mathcal{H}}(x) = |\mathcal{H}|^{-\frac{1}{2}} \mathcal{K}(\mathcal{H}^{-\frac{1}{2}} z)$, where the kernel $\mathcal{K}$ itself is a probability density with zero mean and identity covariance and satisfying* $\lim_{\|x\| \to \infty} \|x\|^d \mathcal{K}(x) = 0$.

Common choices for $\mathcal{K}$ that satisfy the above assumption include Gaussian and Epanechinikov kernels. As an example, Fig. 7.1 visualizes the construction of the standard KDE of 5 data points (black circles) using the well-known Gaussian kernel that is defined as $\mathcal{K}_{\mathcal{H}}(x - x_i) = (\frac{1}{\sqrt{2\pi}h})^d \exp(-\frac{\|x - x_i\|^2}{2h^2})$, where h is the bandwidth. The red curves are the component densities, and each red curve is a scaled version of the normal density curve centered at a datum. The standard KDE is obtained by summing these five scaled components.

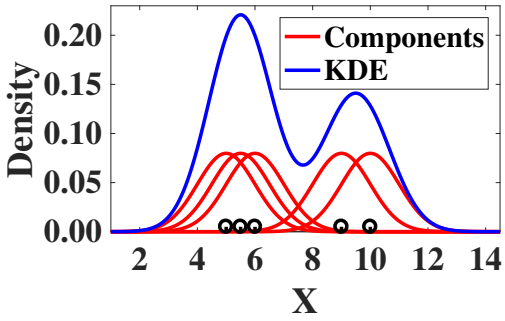


Figure 7.1: An example for the standard KDE

7.4 Methodology

7.4.1 Overview

To achieve the goal described in Section 2, we propose a novel privacy-aware synthesizing method for crowdsourced data (i.e., **PrisCrowd**), which contains two phases. In the first phase, we propose to use the weighted KDE as an intermediate representation of the raw data. This intermediate representation can well capture the statistical properties of the raw data. In the second phase, we first sample a set of candidate synthetic claims from the learned densities in the first phase, then each of these candidate claims is subjected to the proposed privacy test. If the claim passes the privacy test, it will be released, otherwise it will be discarded. The flowchart of the proposed two-phase method is shown in Fig. 7.2.

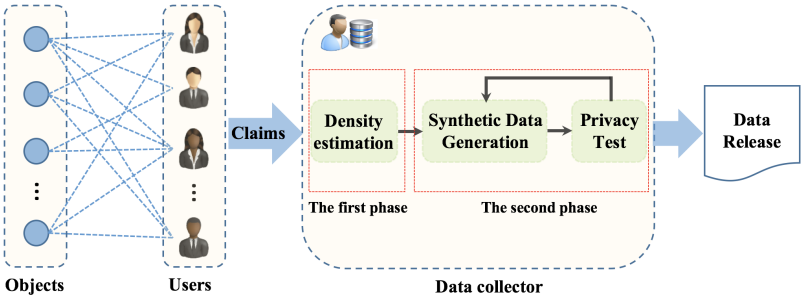


Figure 7.2: Privacy-aware synthesizing for crowdsourced data

7.4.2 Weighted KDE-based Data Representation

In order to share "wealth" data with the data analyzer, the data collector first needs to learn the characteristics or the underlying patterns of original data, i.e., the informative density distributions of objects. To estimate the density for each object, the standard kernel density estimation

(KDE) can be adopted. Additionally, since different users may provide different claims for the same object, the reliability degrees (or weights) of these users should be taken into account when estimating the densities [154, 155, 156, 157]. However, the standard KDE cannot differentiate the importance of users (i.e., user reliability degrees). In order to learn users' reliability and compute the densities of objects simultaneously, we propose a novel method which can estimate users' global and local weights, and then combine them to learn objects' informative density distributions. A user's global weight reflects his capability to provide truthful information for all the objects, and the local weights represent that this user may have different confidence when providing claims for different objects. The advantage of the proposed method is that it can estimate reasonable reliability for each user, and in turn, learn the accurate informative density distributions for objects.

Global Weight Estimation

To evaluate the overall importance of users, the data collector assigns a global weight $g_s \in \mathbb{R}$ to each user s. Meanwhile, we can obtain a global density f_i^g for each object o_i, which should be close to the distribution of claims from reliable users. The distribution of the input claims $\mathcal{X}_i$ can be obtained by $\mathcal{K}_{\mathcal{H}_i}(x, \mathcal{X}_i)$ ($x \in \mathbb{R}$ is a variable), i.e., the kernel function associated with a reproducing kernel Hilbert space $\mathcal{H}_i$. To minimize the weighted deviation from the estimated density $Q = \{f_i^g(x)\}_{i=1}^N$ to the multi-user input $\mathcal{X} = \{\mathcal{X}_i\}_{i=1}^N$, we propose the following optimization framework

$$\min_{G,Q} \sum_{s\in\mathcal{U}} g_s \sum_{i\in\mathcal{E}_s} d_{\mathcal{H}_i}(\mathcal{K}_{\mathcal{H}_i}(x, x_{i,s}), f_i^g(x)) \quad s.t. \ \sum_{s\in\mathcal{U}} \exp(g_s) = 1, \tag{7.1}$$

where $\mathcal{E}_s$ denotes the set of objects observed by user s, $G = \{g_s\}_{s\in\mathcal{U}}$ and the normalized squared loss $d_{\mathcal{H}_i}(\mathcal{K}_{\mathcal{H}_i}(\cdot,\cdot), f_i^g(x))$ is defined as $d_{\mathcal{H}_i}(\mathcal{K}_{\mathcal{H}_i}(x, x_{i,s}), f_i^g(x)) = \|\mathcal{K}_{\mathcal{H}_i}(x, x_{i,s}) - f_i^g(x)\|^2_{\mathcal{H}_i}$. The global loss function (i.e., Eqn. (7.1)) extends the framework in [154] from real space to Hilbert space. We can use an iterative procedure to solve it. Specifically, in the k-th iteration, g_s is updated as

$$g_s^{(k+1)} = -\log \frac{\sum_{i\in\mathcal{E}_s} d_{\mathcal{H}_i}(\mathcal{K}_{\mathcal{H}_i}(x, x_{i,s}), f_i^{g(k)}(x))}{\sum_{s'\in\mathcal{U}} \sum_{i\in\mathcal{E}_{s'}} d_{\mathcal{H}_i}(\mathcal{K}_{\mathcal{H}_i}(x, x_{i,s'}), f_i^{g(k)}(x))}, \tag{7.2}$$

where $f_i^{g(k)}(x) = \sum_{t\in\mathcal{U}_i} g_t^{(k)} \mathcal{K}_{\mathcal{H}_i}(x, x_{i,t}) / (\sum_{t\in\mathcal{U}_i} g_t^{(k)})$. Eqn. (7.2) shows that a user's global weight is inversely proportional to the distance between its claims and the estimated global

densities at the log scale. Users whose claims are close to the derived global densities will have higher global weights.

Local Weight Estimation

As described above, each user may have different confidence when providing claims for different objects. Thus, we need to model the local weight of each user on every object, which will in turn help to infer the accurate density estimations. A potential way to achieve this is to establish a square loss function. However, it leads to a problem that each user would receive the same local weight, and the trustworthiness of the claims provided by different users would be equal. In order to address this problem, we use Hampel loss function [158]:

$$\zeta_{q_1,q_2,q_3}(y) = \begin{cases} y^2/2, & 0 \le y < q_1 \\ q_1 y - q_1^2/2, & q_1 \le y < q_2 \\ \frac{q_1(y-q_3)^2}{2(q_2-q_3)} + \frac{q_1(q_2+q_3-q_1)}{2}, & q_2 \le y < q_3 \\ q_1(q_2+q_3-q_1)/2, & q_3 \le y, \end{cases}$$

where $q_1 < q_2 < q_3$ are predefined nonnegative parameters. These parameters allow us to decrease the trustworthiness of "bad" claims and increase that of "good" ones for each object, so the importance of users can be well distinguished.

Since we incorporate users' reliability into estimating the local densities, the local kernel density of object o_i can be defined as $f_i^l(x) = \sum_{s \in \mathcal{U}_i} l_{i,s} \mathcal{K}_{\mathcal{H}_i}(x, x_{i,s})$, where $l_{i,s}$ is the local weight of the user s on object o_i. Thus, the objective function for estimating $l_i = \{l_{i,s}\}_{s \in \mathcal{U}_i}$ is

$$J(l_i) = \min_{l_i} \sum_{s' \in \mathcal{U}_i} \zeta(\|\mathcal{K}_{\mathcal{H}_i}(x, x_{i,s'}) - \sum_{s \in \mathcal{U}_i} l_{i,s} \mathcal{K}_{\mathcal{H}_i}(x, x_{i,s})\|), \tag{7.3}$$

where $\|\cdot\|$ denotes the difference between users' claims and the estimated local density $f_i^l(x)$. This objective function is not convex, i.e., Eqn. (7.3) does not have a closed form solution. Fortunately, it is possible to approximate $l_i = \{l_{i,s}\}_{s \in \mathcal{U}_i}$ with a standard iteratively re-weighted least squares (IRWLS) algorithm. The iterative procedure for computing $\{l_{i,s}\}_{s \in \mathcal{U}_i}$ is

$$l_{i,s}^{(k+1)} = \frac{\zeta(\|\mathcal{K}_{\mathcal{H}_i}(x, x_{i,s}) - \sum_{t \in \mathcal{U}_i} l_{i,t}^{(k)} \mathcal{K}_{\mathcal{H}_i}(x, x_{i,t})\|)}{\sum_{s' \in \mathcal{U}_i} \zeta(\|\mathcal{K}_{\mathcal{H}_i}(x, x_{i,s'}) - \sum_{t \in \mathcal{U}_i} l_{i,t}^{(k)} \mathcal{K}_{\mathcal{H}_i}(x, x_{i,t})\|)},$$

where k denotes the number of iterations. The above equation shows that the users would receive lower weights when they provide "bad" claims which deviate largely from the center $f_i^l(x) = \sum_{t \in \mathcal{U}_i} l_{i,t} \mathcal{K}_{\mathcal{H}_i}(x, x_{i,t})$.

Combined Weight Estimation

For each user s, to measure the consistency degree of the global and local weights (i.e., g_s and $l_{i,s}$), we define a mixture weight, named combined weight $c_{i,s}$. To learn the combined weight, the relative entropy is employed, which minimizes the information loss between user's global weight and local weight. The smaller the relative entropy value of those weights, the higher the degree of their consistency. The objective of the combined model is

$$\begin{aligned} \min_{\{c_{i,s}\}_{s\in\mathcal{U}_i}} \sum_{s\in\mathcal{U}_i} c_{i,s}\log\frac{c_{i,s}}{l_{i,s}} + \sum_{s\in\mathcal{U}_i} c_{i,s}\log\frac{c_{i,s}}{g_s}. \\ s.t. \sum_{s\in\mathcal{U}_i} c_{i,s} = 1, c_{i,s} \geq 0. \end{aligned} \tag{7.4}$$

By solving Eqn. (7.4), we can obtain the combined weight $c_{i,s}$ of user s on the object o_i as $c_{i,s} = \sqrt{l_{i,s}g_s}/(\sum_{t\in\mathcal{U}_i}\sqrt{l_{i,t}g_t})$. Based on the learned combined weights, we can obtain the density of object o_i which is the weighted sum of claims in Hilbert space and is given as

$$f_i(x) = \sum_{s\in\mathcal{U}_i} \frac{\sqrt{l_{i,s}g_s}}{\sum_{t\in\mathcal{U}_i}\sqrt{l_{i,t}g_t}} \mathcal{K}_{\mathcal{H}_i}(x, x_{i,s}). \tag{7.5}$$

7.4.3 Privacy Test-based Synthetics Release

To provide strong privacy protection for users' private information, in this section, we propose a privacy test-based synthetics release method, which contains two steps: *Candidate synthetics generation* and *Privacy test for candidate synthetics*. In the first step, we sample a set of synthetic claims from the learned density in Eqn. (7.5) as the candidate data to release. Then, in the second step, these sampled synthetics are subjected to a privacy test. If a synthetic claim passes the test, it will be released, otherwise it will be discarded.

Candidate Synthetics Generation

We first discuss how to generate the synthetic claims $\tilde{\mathcal{X}}_i$ for each object o_i. Specifically, we generate each element in $\tilde{\mathcal{X}}_i$ as follows:

1. Select a random integer $s \in \mathcal{U}_i$ with probability $c_{i,s}$;
2. Generate a synthetic claim $\tilde{x}_{i,s}$ by sampling from the probability distribution $\mathcal{K}_{\mathcal{H}_i}(x, x_{i,s})$.

Here $c_{i,s}$ can be treated as the sampling probability that determines whether $x_{i,s}$ is selected or not. In this step, we aim to select some seed data (e.g., $x_{i,s}$) and then probabilistically transform them into the synthetic data. The sampling mechanism used here can increase the uncertainty

of the adversary about whether a user's data is in the released dataset or not, and thus it can help to protect users' privacy to some extent. However, it is not enough to only use the sampling mechanism, directly releasing the sampled data can still violate users' privacy [159]. To tackle this problem, we design the following privacy test mechanism to further prevent users' private information from being disclosed.

Privacy Test for Candidate Synthetics

To prevent an adversary from deducing that a particular claim in $\mathcal{X}_i$ is more responsible for generating the released synthetic data than other claims, the following randomized privacy test mechanism is proposed. Each candidate synthetic data in $\tilde{\mathcal{X}}_i$ is subjected to the randomized privacy test, and it is released only when it passes this test.

Suppose $k \geq 1$ and $\gamma > 1$ are the privacy parameters, and ϵ_0 is the randomness parameter. Let $\mathcal{M}(\cdot)$ denote the above synthetic data generation procedure, which samples a candidate synthetic based on a seed data. Given $x_{i,s} \in \mathcal{X}_i$, we use $\Pr\{\tilde{x}_{i,s} = \mathcal{M}(x_{i,s})\}$ to denote the probability that a synthetic data $\tilde{x}_{i,s}$ is generated based on $\mathcal{M}(\cdot)$. Then the privacy test procedure for $\tilde{x}_{i,s}$ is described as follows:

1. Randomize k by adding a noise: $\tilde{k} = k + z$, where $z \sim Lap(1/\epsilon_0)$ is sampled from the Laplace Distribution.
2. Let $a \geq 0$ be the integer that satisfies the inequalities $\gamma^{-a-1} < \Pr\{\tilde{x}_{i,s} = \mathcal{M}(x_{i,s})\} \leq \gamma^{-a}$.
3. Let k' be the number of records $x_{i,s'} \in \mathcal{X}_i$ that satisfies $\gamma^{-a-1} < \Pr\{\tilde{x}_{i,s} = \mathcal{M}(x_{i,s'}\} \leq \gamma^{-a}$.
4. If $k' \geq \tilde{k}$, $\tilde{x}_{i,s}$ passes the test, otherwise it fails.

Note that k' denotes the number of possible data seeds that can generate $\tilde{x}_{i,s}$ with a probability value falling into a very stringent interval $[\gamma^{-a-1}, \gamma^{-a}]$. The threshold parameter $\tilde{k}$ prevents releasing sensitive synthetic data. Under this randomized privacy test, a candidate synthetic data is released only when there are at least $\tilde{k}$ possible data seeds that can generate $\tilde{x}_{i,s}$. Intuitively, the larger the value of k, the larger the number of the possible seed data that are indistinguishable from $x_{i,s}$. Also, the less the value of γ, the more difficult to distinguish $x_{i,s}$ from other possible seed data. Algorithm 6 summarizes the proposed privacy test-based synthetics release procedure, in which m denotes the number of synthetic claims that need to be released for object o_i.

Algorithm 6 Private test-based synthetics release for o_i

Input: $\{c_{i,s}\}_{s\in\mathcal{U}_i}$, $\mathcal{X}_i = \{x_{i,s}\}_{s\in\mathcal{U}_i}$, k, γ, ϵ_0, and m.
Output: The dataset $\tilde{\mathcal{X}}_i$ that can be released.

$\tilde{\mathcal{X}}_i = \emptyset$
while $|\tilde{\mathcal{X}}_i| < m$ **do**
  Select a random integer $s \in \mathcal{U}_i$ with probability $c_{i,s}$;
  Generate a synthetic claim $\tilde{x}_{i,s}$ based on the probability distribution $\mathcal{K}_{\mathcal{H}_i}(x, x_{i,s})$;
  Conduct randomized privacy test for $\tilde{x}_{i,s}$;
  if $\tilde{x}_{i,s}$ passes the privacy test **then**
    $\tilde{\mathcal{X}}_i = \tilde{\mathcal{X}}_i \bigcup \{\tilde{x}_{i,s}\}$;
  end if
end while
return $\tilde{\mathcal{X}}_i$;

7.4.4 Theoretical Analysis

Consistency Analysis

In Section 4.3, we generate the synthetic claims for object o_i by sampling from the mixture distribution $f_i(x)$, i.e., $\tilde{x}_{i,s} \sim f_i(x)$. After obtaining the dataset $\tilde{\mathcal{X}}_i = \{\tilde{x}_{i,s}\}_{s=1}^m$, a basic question here is that how well the generated dataset can reflect the original density function $f_i(x)$. Since each $\tilde{x}_{i,s}$ is sampled from $f_i(x)$ independently, the density function over $\{\tilde{x}_{i,s}\}_{s=1}^m$ can be denoted as $\tilde{f}_i(x) = \frac{1}{m}\sum_{s=1}^m \mathcal{K}_{\mathcal{H}_i}(x, \tilde{x}_{i,s})$. In Theorem 1, we provide the expected squared L_2-norm distance between $f_i(x)$ and $\tilde{f}_i(x)$.

Theorem 11. *Under Assumption 1 for $\mathcal{K}_{\mathcal{H}_i}$ with the diagonal bandwidth matrix $\mathcal{H}_i = \hat{h}^2 I_d$, we further assume that the support of $\mathcal{K}(z)$ satisfies $\|z\| \leq 1$. Then, the expected squared L_2-norm distance between $f_i(x)$ and $\tilde{f}_i(x)$, i.e., $J = \mathbb{E}[\int (f_i(x) - \tilde{f}_i(x))^2 dx]$, satisfies*

$$J \leq 4A\hat{h} + A^2\hat{h}^2 V + \frac{B}{m\hat{h}^d} + \frac{ABV}{m\hat{h}^{d-1}}, \tag{7.6}$$

where $A = \sup_{x\in\mathbb{R}^d} \|\nabla f_i(x)\|$, $B = \int (\mathcal{K}(z))^2 dz$ and V is the volume of the support of $f_i(x)$. The expectation is respected to $\{\tilde{x}_{i,s}\}_{s=1}^m \sim f_i(x)$. This theorem is a general result for d dimensional case, in this chapter, the value of d is 1.

Privacy Analysis

Next, we conduct privacy analysis for Algorithm 1. Based on Theorem 2, we know that the proposed algorithm is differentially private.

Theorem 12. *Note that the input parameters of Algorithm 1 include* $\{c_{i,s}\}_{s\in\mathcal{U}_i}$, $k \geq 1$, $\gamma > 0$, *and* ϵ_0. *For any neighboring datasets* $\mathcal{X}_i$ *and* $\mathcal{X'}_i$ *such that* $|\mathcal{X}_i|, |\mathcal{X'}_i| \geq k$ *and any integer* $1 \leq t < k$, *we have that Algorithm 1 is* (ϵ, δ)*-differentially private, where* $\epsilon = \epsilon_0 + \log(1 + \frac{\gamma}{t}\frac{\max_{s\in\mathcal{U}_i} c_{i,s}}{\min_{s\in\mathcal{U}_i} c_{i,s}})$, $\delta = |\mathcal{U}_i| \max_{s\in\mathcal{U}_i} c_{i,s} e^{-\epsilon(k-t)}$.

Remark 3. *Note that the proposed Algorithm 6 is different from the mechanism in [160]. The probability of choosing the seed* $x_{i,s}$ *is non-uniform in Algorithm 6 while that is uniform in [160]. The non-uniform property may generate different parameters of differential privacy. When* $\max_{s\in\mathcal{U}_i} c_{i,s} = \min_{s\in\mathcal{U}_i} c_{i,s} = 1/|\mathcal{U}_i|$ *(i.e., we uniformly sample the seed* $x_{i,s}$*), the above Theorem 12 is actually Theorem 1 in [160]. Thus, Theorem 12 in in this chapter is a generalization of Theorem 1 in [160]. Although the main idea of the proof for Theorem 2 is similar to that in [160], the details are quite different: in [160] the proof consider* $\mathcal{X'}_i = \mathcal{X}_i \bigcup \{x_{i,s'}\}$ *as the neighborhood dataset while ours consider* $\mathcal{X'}_i = \{\mathcal{X}_i - \{x_{i,s}\}\} \bigcup \{x_{i,s'}\}$ *as the neighborhood dataset. That is because if we add one data record, the probability of sampling seeds, i.e.,* $\{c_{i,s}\}$*, will be totally changed. So the proof in [160] cannot satisfy our case.*

7.5 Experiments

Performance measure. To evaluate the performance of our method, we adopt two measure metrics: *the integrated squared error (ISE)* and *the squared integrated squared error (SISE)*. *ISE* is defined as: $\sum_{i=1}^{N} \int_{-\infty}^{+\infty} (f_i - \tilde{f}_i)^2 dx$, where f_i and $\tilde{f}_i$ are respectively the original density and the density derived from the synthetic data for object o_i. *SISE* is defined as: $\sum_{i=1}^{N} (\int_{-\infty}^{+\infty} (f_i - \tilde{f}_i)^2 dx)^2$. Compared with *ISE*, *SISE* tends to penalize more on the large distance and less on the small distance. Since the goal of the collector is to release the data whose pattern is similar to the true underlying pattern for the objects, the lower the *ISE* or *SISE*, the better the method.

Datasets. We adopt the following datasets for our experiments: *Population Dataset* [161, 162], *Stock Dataset* [163], and *Indoor Floorplan Dataset* [155]. The statistics of these real-world datasets are provided in Table 7.1.

Dataset	# users	# objects
Population	2,344	1,124
Stock	55	5,521
Indoor Floorplan	247	129

Table 7.1: The statistics of the adopted datasets.

Baselines. Here, we adopt two baselines, i.e. *Basic* and *Uniform*. In the *Basic* method, the data collector adds three level noise to the original data: $\epsilon = 0.1$ (Strong), $\epsilon = 1$ (normal) and $\epsilon = 10$ (Weak). In the *Uniform* method, the collector treats all users equally and the entities' densities are learned with the uniformly weighted kernel density estimation. Here, the synthetic data generation and the privacy tests procedures are the same with those in our proposed method.

Case study. In order to investigate the advantages of the users' combined weights, we conduct case studies on the three real-world datasets. For each dataset, we randomly select two objects as the cases, and then estimate their densities. The estimated densities are shown in Fig. 7.3. The red line in each subfigure represents the density estimated based on users' combined weights. The black line represents the result estimated only based on the global weight of each user. We also conduct estimations without considering user quality, i.e., treating all users equally, and the estimated density for each object is represented with the green line. The results in Fig.3 show that the densities estimated based on users' combined weights are the closest to the true densities which are represented with the blue lines. Additionally, we show the claims of each object in this figure with magenta circles and crosses. We can see that some claims (i.e., the magenta crosses) are far away from others (i.e., the magenta circles). These claims are usually provided by the users with low weights, and they can be treated as outliers when estimating each object's density. The results show that the method based on the combined weight is more robust to outliers than the methods which only adopt users' global weights or treat all users equally. In other words, the estimated density for each object based PrisCrowd can well reflect the underlying true density of this object.

Accuracy comparison. In this experiment, we evaluate the accuracy (or quality) of the published synthetic data and explore whether these data can well reflect the underlying true densities of the objects. Here we assume that the data collector releases 30 synthetic claims for each object to the public. The parameters γ and k are set as 4 and 5 respectively. In order to evaluate the accuracy of the synthetic claims, we first derive the density (i.e., $\tilde{f}_i$) of each object from

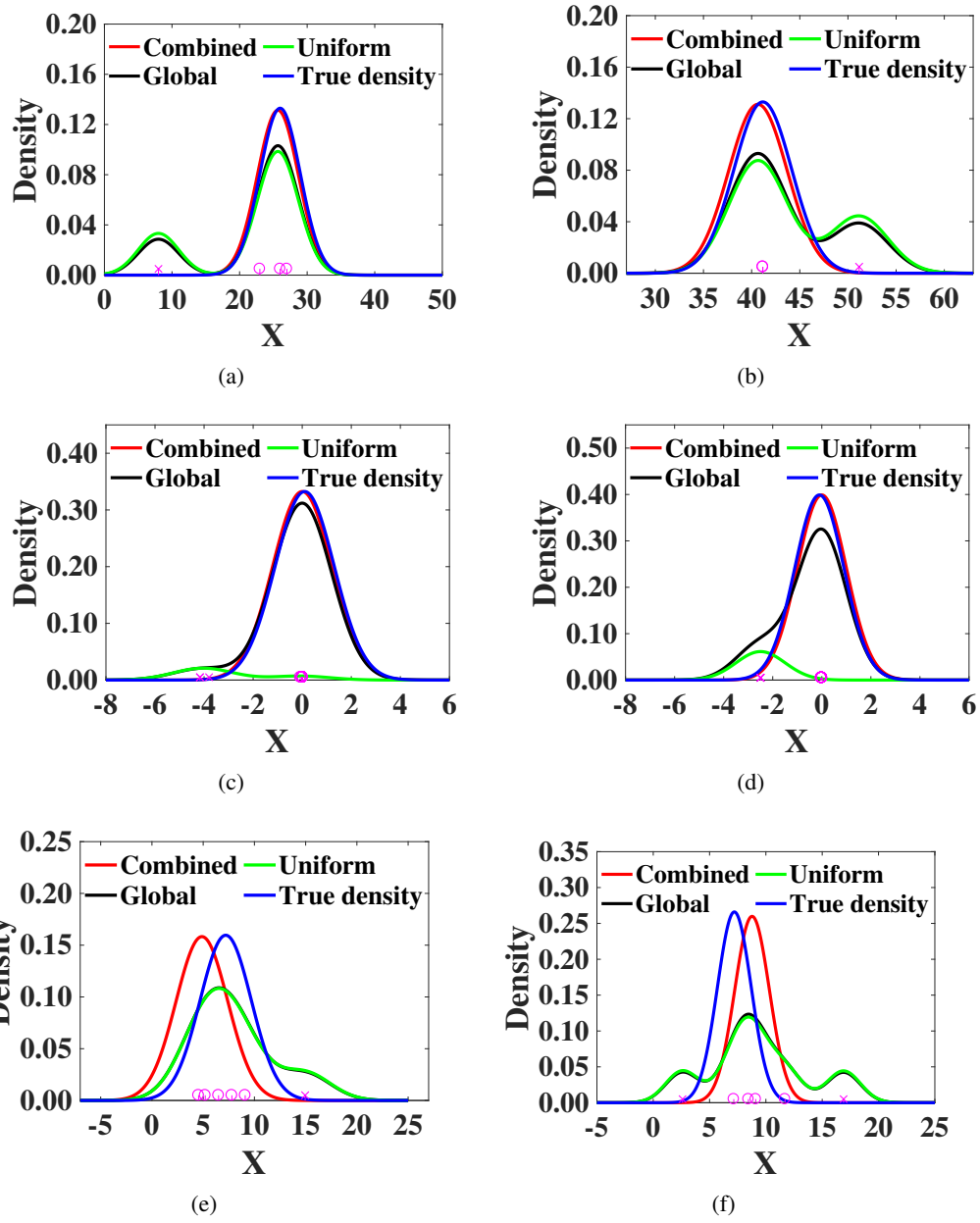


Figure 7.3: Case study on real-world datasets. (a) and (b): the two cases for Population dataset. (c) and (d): the two cases for Stock dataset. (e) and (f): the two cases for Indoor Floorplan dataset.

the synthetic data, and then calculate *ISE* and *SISE* for each dataset. The results are shown in

Measure	Method	Population	Stock	Indoor
ISE	**PrisCrowd**	**0.479**	**1.699**	**1.051**
	Uniform	0.628	17.628	1.220
	Basic(Strong)	1.183	12.799	1.943
	Basic(Normal)	1.119	9.867	1.937
	Basic(Weak)	0.866	2.125	1.882
SISE	**PrisCrowd**	**6.209**	**11.430**	**8.405**
	Uniform	8.502	47.217	11.112
	Basic(Strong)	12.013	40.420	15.111
	Basic(Normal)	11.768	35.391	15.046
	Basic(Weak)	10.149	15.412	14.723

Table 7.2: Accuracy comparison on the real-world datasets

Table 7.2, from which we can see the proposed approach performs much better than the baseline methods on all real-world datasets. That is to say, the synthetic data generated based on our proposed method could well preserve the characteristics of the underlying pattern for the objects. Additionally, the results also show that the advantages of our proposed approach on the Stock dataset is larger than that on the Population and Indoor Floorplan datasets. The reason is that there are more outlying data points in the Stock dataset, and our proposed approach is robust to these outliers while the baseline methods are very sensitive to them.

The effect of the number of sampled claims. In this experiment, we evaluate the effect of the number of sampled claims for each object on the performance of the proposed method. Here we vary the number of the sampled claims for each object from 1 to 30 and then calculate the *ISE* and *SISE* on the three real-world datasets. The results are shown in Fig. 7.4, from which we can see the *ISE* and *SISE* gradually get flattened with the increase of the number of the sampled claims for each object. Take the population dataset as an example, when the number of sampled claims is lager than 10, the values of *ISE* and *SISE* are almost the same. That is to say, the released data generated based on our proposed method could well reflect the underlying patterns of the objects even only a few claims are sampled for each object.

Computational cost. Next we evaluate the computational cost of the synthetic claims generation procedure, i.e., the second phase in our proposed method. In this experiment, we only

generate synthetic claims for the objects whose ground truths can be achieved from the original datasets, and consider two scenarios, i.e., with privacy tests and without privacy tests. Then we vary the number of the sampled claims for each object from 1 to 30. The running time of

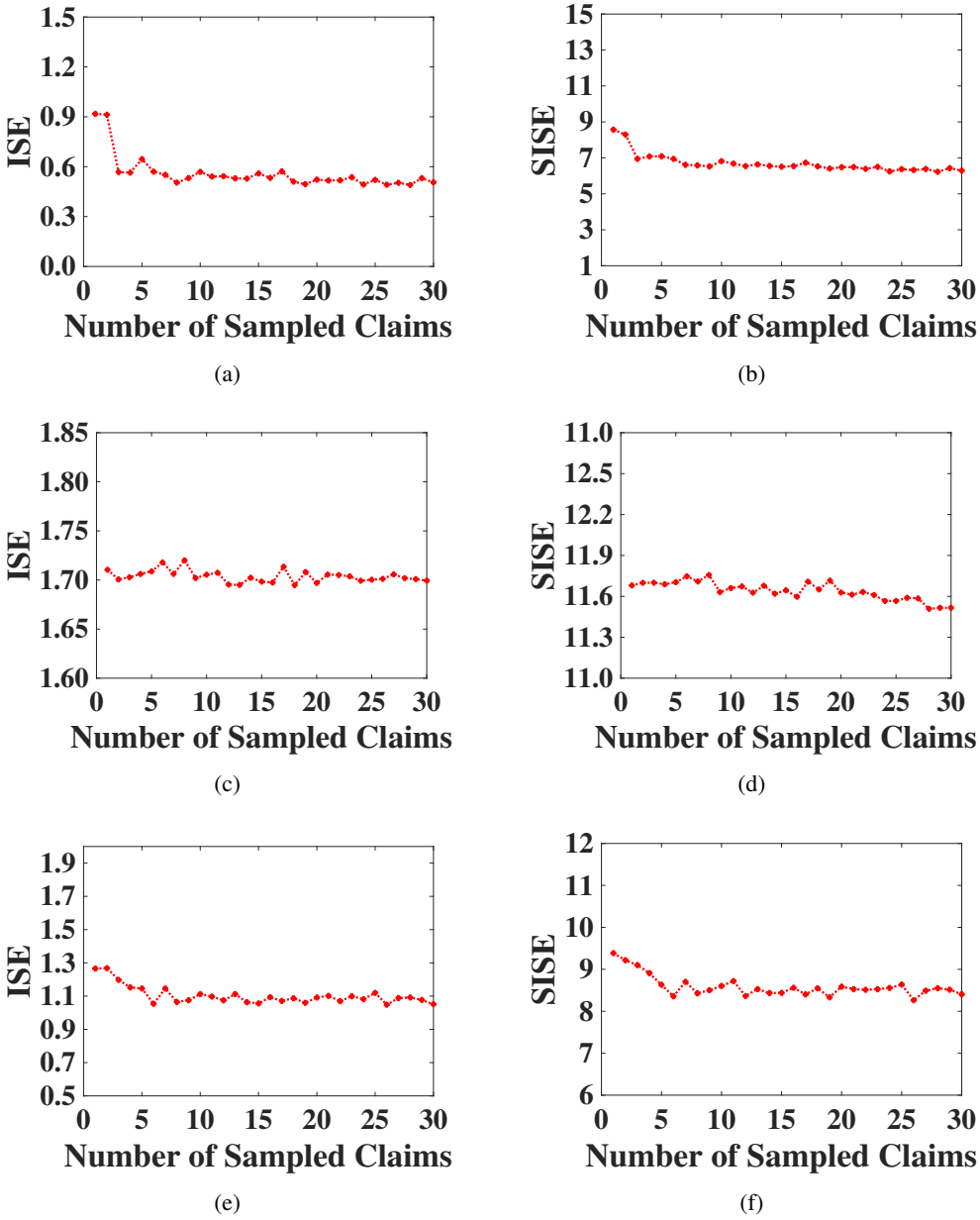
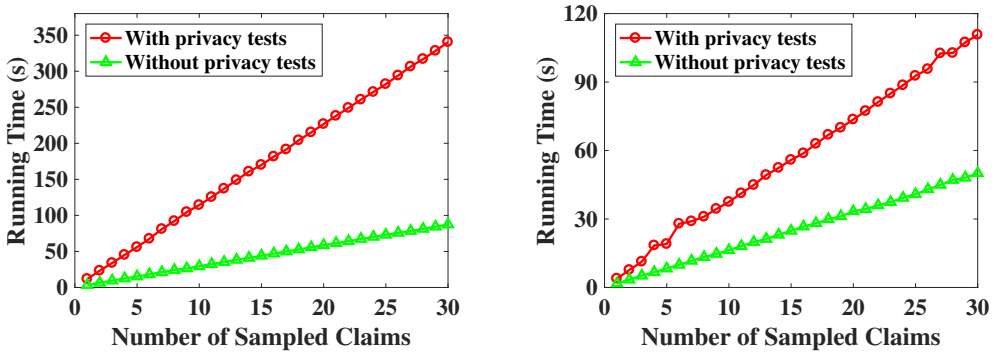


Figure 7.4: Accuracy w.r.t. Number of Sampled Claims. (a) and (b): Population. (c) and (d): Stock. (e) and (f): Indoor Floorplan.



(a) (b)

Figure 7.5: Running time vs. number of sampled claims for each object. (a): Population. (b): Indoor Floorplan.

the synthetic claims generation procedure for the Population and Indoor Floorplan datasets is shown in Fig. 7.5, from which we can see the running time in the two scenarios is approximately linear with respect to the number of sampled claims for each object. Additionally, the results also show that the privacy test step introduce extra computational cost during the released data generation procedure. This is because each candidate synthetic data record needs to be tested in the privacy test step. Since good utility can be achieved based on our proposed method even only a few synthetic claims are generated for each object, the computational cost is tolerable in practice.

7.6 Related Work

Recently, various differential private data release approaches have been proposed. Those methods can be roughly partitioned into two categories: the interactive ones and the non-interactive ones. In an interactive method [164, 165, 166], a data analyzer can pose queries via a private mechanism, and a dataset owner answers these queries in response. In the non-interactive framework [167, 168, 169, 170, 171], a data owner releases the private version of the original data. Once data are published, the owner has no further control over the published data.

The method here is non-interactive. The typical approach to protect data privacy in the non-interactive context is to directly add noise, which is taken by [168, 169]. These works are either computationally infeasible on high-dimensional data, or practically ineffective because of their large utility costs. There are also some other works [168] which release private data without

adding noise, but they are unsuitable to be used in the newly appearing crowdsourcing setting considered in this chapter where multi-sources provide multi-observations for multi-objects.

7.7 Conclusions

In this chapter, we propose a novel privacy-aware synthesizing method for crowdsourced data. Based on this method, the data collector can release the crowdsourced data with strong privacy protection for users' private information, while at the same time, the data analyzer can achieve good utility from the released data. Both theoretical analysis and extensive experiments on real-world datasets verify the effectiveness of the proposed method.

Chapter 8

Conclusions and Future Directions

In this dissertation, the essential characteristics at the core of trustworthy machine learning (i.e., model transparency, security vulnerability and robustness, and privacy preservation) are studied. Without fully studying the trustworthiness of the deployed machine learning-based systems, we will face a variety of devastating societal and environmental consequences. For example, attackers may actively manipulate the perceptual systems of autonomous vehicles, which can potentially lead to a multitude of disastrous consequences, ranging from a life-threatening accident to even a large-scale interruption of transportation services relying on autonomous cars.

On the model transparency aspect, we develop several effective post-hoc interpretation methods to provide the feature-level explanations for pairwise models, which aim to model the relative relationship between pairs of samples. Based on the designed pairwise interpretation methods, we can investigate how the input features contribute to model predictions for patient similarity learning. In addition, we also interpret the generalization ability of deep metric learning models to generalize to unseen data by presenting the generalization error bound for deep metric learning.

On the security aspect, we first investigate the vulnerability of DRL interpretations to malicious attacks. Specifically, we first propose a universal adversarial attack against DRL interpretations, which aims to craft a single universal perturbation that can be applied identically on every time step. Based on this attack, the attacker can efficiently deceive downstream DRL interpretation methods via state perturbations. Then, we design a model poisoning attack against DRL interpretations, based on which the attacker can secretly alter the interpretation results through providing the agent a strategically poisoned but equally effective pretrained DRL model. In addition, to reduce the susceptibility of the generated concept-based explanations to

adversarial attacks, we also design a novel robust and automatic interpretation method, which can not only automatically provide the prototype-based concept explanations but also provide certified robustness guarantees for the generated prototype-based explanations.

On the privacy aspect, we first propose several differentially private pairwise learning algorithms for both online and offline settings. Specifically, for the online setting, I first proposed a differentially private algorithm for the strongly convex loss functions, and then extend this algorithm to general convex loss functions by proposing another differentially private algorithm. For the offline setting, I also proposed differentially private algorithms for strongly convex loss functions and general convex loss functions, and then derived their regret upper bounds. In addition, we also design a novel privacy-aware synthesizing method for crowdsourced data, which can not only extract high quality crowdsourced data via differentiating each user's fine grained reliability degrees on different objects but also achieve differential privacy without injecting noise to the data.

The development of trustworthy machine learning systems can inspire user trust in the system's trustworthiness compliance, while users' trust in a machine learning system can assist in achieving a system's trustworthy objectives by helping users to take appropriate responses to untrustworthy issues and to avoid incidental actions. However, existing research works on trustworthy machine learning usually only focus on one aspect, and fail to take into consideration other untrustworthy aspects. For example, by using existing works on counterfactual explanations, we can explain a prediction by calculating a change in a datapoint that would cause the underlying machine learning model to classify it in a desired class. However, releasing additional information is a risky prospect from a privacy perspective. The generated counterfactual explanations, as functions of the model trained on a private dataset, might inadvertently leak information about the training set, beyond what is necessary to provide useful counterfactual explanations. In fact, an adversary can leverage model explanations to perform different privacy attacks to infer private information about the training data. Despite this potential privacy risk, there has been little effort to analyze and address the data privacy concerns that might arise due to the release of model explanations. Therefore, one future research direction is about new ways to design the trustworthy machine learning systems that are trustworthy in all of the desired trust aspects. Another future research direction is to devise more sophisticated and effective mechanisms to automate the process of designing trustworthy machine learning systems. In practice, humans are heavily involved in almost every process of building trustworthy machine learning systems (e.g., model and hyper-parameter selection). For example, existing perturbation-based robustness techniques usually need particular manual derivations and implementations when

extended to different architectures. Thus, to make trustworthy machine learning systems easier to apply and reduce the demand for experienced human experts, it is important for us to work on the automatic design of trustworthy machine learning systems.

Chapter 9

Appendix

9.1 Proof of Theorem 1

Proof. The proof of this theorem contains two steps. In the first step, we will verify that the importance values (i.e., $\{\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)\}_{d=1}^{D}$) calculated by the proposed ASIPair satisfy the four properties: efficiency, fairness, dummy and additivity. In the second step, we will show that there exists exactly one solution that has all the four properties.

Step 1: First of all, let us consider the efficiency property. The summation $\sum_{d=1}^{D} \phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)$ is calculated as follows

$$\begin{aligned}\sum_{d=1}^{D} \phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) &= \sum_{d=1}^{D} \frac{1}{D!} \sum_{\mathcal{O} \in \pi(D)} \{\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{P_{\mathcal{O}}^d \cup \{d\}}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d \cup \{d\}})] - \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{P_{\mathcal{O}}^d}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d})]\} \\ &= \frac{1}{D!} \sum_{\mathcal{O} \in \pi(D)} \sum_{d=1}^{D} \{\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{P_{\mathcal{O}}^d \cup \{d\}}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d \cup \{d\}})] - \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{P_{\mathcal{O}}^d}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d})]\} \\ &= \frac{1}{D!} \sum_{\mathcal{O} \in \pi(D)} \zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \zeta(\boldsymbol{x}_i, \boldsymbol{x}_j), \end{aligned} \tag{9.1}$$

where $\pi(D)$ denotes the set of of all possible ordered permutations of the feature indices $\{1, 2, ..., D\}$, and $\mathcal{O} \in \pi(D)$ is a random permutation. The last equality is derived based on that $\phi_{\emptyset}(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) = 0$.

Secondly, we will verify the fairness property. The difference $\phi_{d_1}(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) - \phi_{d_2}(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)$

is calculated as follows

$$\begin{aligned}
&\phi_{d_1}(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) - \phi_{d_2}(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) \\
&= \sum_{T \subset [D] \setminus \{d_1\}} \frac{|T|!(D-|T|-1)!}{D!} \{\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{T \cup \{d_1\}}, \boldsymbol{x}_j^{T \cup \{d_1\}}] - \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{T}, \boldsymbol{x}_j^{T}]\} - \\
&\quad \sum_{T \subset [D] \setminus \{d_2\}} \frac{|T|!(D-|T|-1)!}{D!} \{\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{T \cup \{d_2\}}, \boldsymbol{x}_j^{T \cup \{d_2\}}] - \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{T}, \boldsymbol{x}_j^{T}]\} \\
&\quad = \sum_{T \subset [D] \setminus \{d_1\}} \frac{|T|!(D-|T|-1)!}{D!} \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{T \cup \{d_1\}}, \boldsymbol{x}_j^{T \cup \{d_1\}}] \\
&\quad - \sum_{T \subset [D] \setminus \{d_2\}} \frac{|T|!(D-|T|-1)!}{D!} \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{T \cup \{d_2\}}, \boldsymbol{x}_j^{T \cup \{d_2\}}] = 0, \qquad (9.2)
\end{aligned}$$

where the first equality is derived based on the definition of $\phi_{d_1}(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)$ and $\phi_{d_2}(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)$. In the above, the last equality is derived based on that for all $T \subset \{1, 2, \cdots, D\} \setminus \{d_1, d_2\}$, $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{T \cup \{d_1\}}, \boldsymbol{x}_j^{T \cup \{d_1\}}] = \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{T \cup \{d_2\}}, \boldsymbol{x}_j^{T \cup \{d_2\}}]$.

Next, we will verify the dummy property. By the definition of $\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)$, we have

$$\begin{aligned}
&\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) = \sum_{T \subset [D] \setminus \{d\}} \frac{|T|!(D-|T|-1)!}{D!} \Delta_d(\boldsymbol{x}_i, \boldsymbol{x}_j, T, \zeta) = \\
&\sum_{T \subset [D] \setminus \{d\}} \frac{|T|!(D-|T|-1)!}{D!} \{\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{T \cup \{d\}}, \boldsymbol{x}_j^{T \cup \{d\}}] - \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | \boldsymbol{x}_i^{T}, \boldsymbol{x}_j^{T}]\} \\
&\quad = \sum_{T \subset [D] \setminus \{d\}} \frac{|T|!(D-|T|-1)!}{D!} * 0 = 0, \qquad (9.3)
\end{aligned}$$

where the third equality is derived based on that for all subset $T \subset \{1, 2, \cdots, D\} \setminus \{d\}$, $\mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | (\boldsymbol{x}_i^{T \cup \{d\}}, \boldsymbol{x}_j^{T \cup \{d\}})] = \mathbb{E}[\zeta(\boldsymbol{x}_i, \boldsymbol{x}_j) | (\boldsymbol{x}_i^{T}, \boldsymbol{x}_j^{T})]$.

Finally, we will prove the additivity property. For $\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta_1+\zeta_2)$, we have the following

$$
\begin{aligned}
&\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta_1 + \zeta_2) \\
&= \frac{1}{D!} \sum_{\mathcal{O}\in\pi(D)} (\mathbb{E}[(\zeta_1 + \zeta_2)(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d \cup \{d\}}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d \cup \{d\}}] - \mathbb{E}[(\zeta_1 + \zeta_2)(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d}]) \\
&= \frac{1}{D!} \sum_{\mathcal{O}\in\pi(D)} \{\mathbb{E}[\zeta_1(\boldsymbol{x}_i, \boldsymbol{x}_j) + \zeta_2(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d \cup \{d\}}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d \cup \{d\}}] - \mathbb{E}[\zeta_1(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&\quad + \zeta_2(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d}]\} \\
&= \frac{1}{D!} \sum_{\mathcal{O}\in\pi(D)} \mathbb{E}[\zeta_1(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d \cup \{d\}}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d \cup \{d\}}] + \frac{1}{D!} \sum_{\mathcal{O}\in\pi(D)} \mathbb{E}[\zeta_2(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d \cup \{d\}}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d \cup \{d\}} \\
&\quad - \{\frac{1}{D!} \sum_{\mathcal{O}\in\pi(D)} \mathbb{E}[\zeta_1(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d}] + \frac{1}{D!} \sum_{\mathcal{O}\in\pi(D)} \mathbb{E}[\zeta_2(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d}]\} \\
&= \{\frac{1}{D!} \sum_{\mathcal{O}\in\pi(D)} (\mathbb{E}[\zeta_1(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d \cup \{d\}}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d \cup \{d\}}] - \mathbb{E}[\zeta_1(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d}])\} \\
&\quad + \{\frac{1}{D!} \sum_{\mathcal{O}\in\pi(D)} (\mathbb{E}[\zeta_2(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d \cup \{d\}}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d \cup \{d\}}] \\
&\quad - \mathbb{E}[\zeta_2(\boldsymbol{x}_i, \boldsymbol{x}_j)|\boldsymbol{x}_i^{P_{\mathcal{O}}^d}, \boldsymbol{x}_j^{P_{\mathcal{O}}^d}])\} = \phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta_1) + \phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta_2),
\end{aligned} \tag{9.4}
$$

where the second equality is derived based on the definition that $(\zeta_1+\zeta_2)(\boldsymbol{x}_i, \boldsymbol{x}_j) = \zeta_1(\boldsymbol{x}_i, \boldsymbol{x}_j)+\zeta_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for any $(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and the last second equality is derived based on the definition of $\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta_1)$ and $\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta_2)$.

Step 2: In this step, we provide a proof sketch to show there exists a unique solution that satisfies the efficiency, fairness, dummy and additivity properties. The uniqueness result can be derived from the fact that the class of games with D features forms a 2^{D-1} dimensional vector space where the set of unanimity games constitutes a basis. For every non-empty $T \subset [D]$, we define η^T (the unanimity game of T) by $\eta^T([D]) = 1$ if $T \subset [D]$, otherwise, $\eta^T([D]) = 0$. It is clear that the dummy and fairness properties together yield a value that is uniquely determined on unanimity games. Combined with the additivity axiom and the fact the unanimity games constitute a basis for the vector space of games, this yields the uniqueness result. □

9.2 Proof of Theorem 2

Proof. Note that $\mathbb{E}[Y_{m,h}] = \sum_{d=1}^{D} \boldsymbol{A}_{m,d}\mathbb{E}[\Delta_d^h(\boldsymbol{x}_i, \boldsymbol{x}_j, P_{\mathcal{O}_h}^d, \zeta)] = \sum_{d=1}^{D} \boldsymbol{A}_{m,d}\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)$, where the second equality is derived based on that $\phi_d(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta) = \mathbb{E}[\Delta_d^h(\boldsymbol{x}_i, \boldsymbol{x}_j, P_{\mathcal{O}_h}^d, \zeta)]$. In

fact, for $Y_{m,h}$, based on the super-additivity, we can then derive that $-r/\sqrt{M} \leq Y_{m,h} = \sum_{d=1}^{D} \boldsymbol{A}_{m,d}\Delta_d^h(\boldsymbol{x}_i, \boldsymbol{x}_j, P^d_{\mathcal{O}_h}, \zeta) \leq +r/\sqrt{M}$. Then, we have the following

$$
\begin{aligned}
P[\|\boldsymbol{A}\boldsymbol{\phi} - \boldsymbol{b}\|_{\mathcal{L}_2} \geq \epsilon] &\leq P[\|\boldsymbol{A}\boldsymbol{\phi} - \boldsymbol{b}\|_{\mathcal{L}_\infty} \geq \frac{\epsilon}{\sqrt{M}}] = P[\max_{m=1,\cdots,M} |\boldsymbol{A}_m\boldsymbol{\phi} - b_m| \geq \frac{\epsilon}{\sqrt{M}}] \\
&\leq P[\cup_{m=1,\cdots,M} |\boldsymbol{A}_m\boldsymbol{\phi} - b_m| \geq \frac{\epsilon}{\sqrt{M}}] \leq \sum_{m=1}^{M} P[|\boldsymbol{A}_m\boldsymbol{\phi} - b_m| \geq \frac{\epsilon}{\sqrt{M}}] \\
&\leq 2M\exp(-\frac{2H^2(\epsilon/\sqrt{M})^2}{\sum_{h=1}^{H}(2r/\sqrt{M})^2}) = 2M\exp(-\frac{H\epsilon^2}{2r^2}),
\end{aligned} \tag{9.5}
$$

where $b_m = \frac{1}{H}\sum_{h=1}^{H} Y_{m,h}$, $\boldsymbol{A}_m$ denotes the m-th row of the measurement matrix $\boldsymbol{A} \in \mathbb{R}^{M\times D}$, $\boldsymbol{\phi} = (\phi_1(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta), \cdots, \phi_D(\boldsymbol{x}_i, \boldsymbol{x}_j, \zeta)) \in \mathbb{R}^D$ and $\boldsymbol{b} \in \mathbb{R}^D$ is the measurement vector. The first inequality is derived based on the fact that $\|\boldsymbol{A}\boldsymbol{\phi} - \boldsymbol{b}\|_{\mathcal{L}_2} \leq \sqrt{N}\|\boldsymbol{A}\boldsymbol{\phi} - \boldsymbol{b}\|_{\mathcal{L}_\infty}$. The second inequality follows the union bound, and the last inequality is derived based on the Chernoff-Hoeffding inequality. Since $\boldsymbol{\phi} = \bar{\phi}\boldsymbol{I}_D + \tilde{\boldsymbol{\phi}}$, we can also derive that the probability that $\|A(\bar{\phi}\boldsymbol{I}_D + \tilde{\boldsymbol{\phi}}) - b\|_{\mathcal{L}_2} \geq \epsilon$ is no greater than $2M\exp(-\frac{H\epsilon^2}{2r^2})$. Now, we want the term $2M\exp(-\frac{H\epsilon^2}{2r^2})$ in Eqn. (9.5) to be bounded by $\delta/2$, we then have the following

$$
2M\exp(-\frac{H\epsilon^2}{2r^2}) \leq \frac{\delta}{2} \Longrightarrow \frac{2r^2}{\epsilon^2}\log\frac{4M}{\delta} \leq H, \tag{9.6}
$$

where H denotes the number of permutations. Thus, based on the above, with probability $(1 - \delta/2)$, we have $\|\boldsymbol{A}\boldsymbol{\phi} - \boldsymbol{b}\|_{\mathcal{L}_2} \leq \epsilon$. In the following, we will proceed to bound the number of measurements M.

The main condition on the measurement matrix $\boldsymbol{A}$ that ensures the exact recovery of the s-sparse parameter from an underdetermined linear system is the restricted isometry condition. For the measurement matrix $\boldsymbol{A} \in \mathbb{R}^{M\times D}$, its s-restricted isometry constant δ_s is defined as the smallest quantity such that the matrix $\boldsymbol{A}_T$ obeys $(1 - \delta_s)\|\boldsymbol{x}\|^2_{\mathcal{L}_2} \leq \|\boldsymbol{A}_T\boldsymbol{x}\|^2_{\mathcal{L}_2} \leq (1 + \delta_s)\|\boldsymbol{x}\|^2_{\mathcal{L}_2}$ for each subset $T \subset [D]$ and all $\boldsymbol{x} \in \mathbb{R}^{|T|}$, where $\boldsymbol{A}_T$ is a $M \times |T|$ matrix containing the columns of $\boldsymbol{A}$ corresponding to T. Based on Theorem 2.7 and 2.12 in [172], we can derive that if $M \geq C(0.465)^{-2}(2s\log(D/(2s)) + \log(2/\delta))$, the restricted isometry constant δ_{2s} of the measurement matrix $\boldsymbol{A}$ satisfies $\delta_{2s} < \frac{3}{4+\sqrt{6}} \approx 0.465$ with probability $1 - \delta/2$. Note that C is a universal constant. Finally, we can derive the following

$$
\|\tilde{\boldsymbol{\phi}} - \boldsymbol{\phi}\|_{\mathcal{L}_2} = \|\hat{\boldsymbol{\phi}}^* - \hat{\boldsymbol{\phi}}\|_{\mathcal{L}_2} \leq \Phi_1\epsilon + \Phi_2\frac{\sigma_s(\boldsymbol{\phi})_{\mathcal{L}_1}}{\sqrt{s}}, \tag{9.7}
$$

where $\sigma_s(\boldsymbol{\phi})_{\mathcal{L}_1} := \inf\{\|\boldsymbol{\phi} - \Psi\|_{\mathcal{L}_1}, \Psi \text{ is } s\text{-sparse}\}$, and Φ_1 and Φ_2 are two constants that depend only on δ_s. Note that $\hat{\phi}^*$ is the solution of the optimization problem for the proposed RAIPair.

□

9.3 Proof of Theorem 3

Proof. Here, we provide a sketch of the proof for this theorem. For simplicity, we write $g(1 + y_{ij}(D(f^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho}), f^L(\boldsymbol{x}_j, \boldsymbol{W}, \boldsymbol{\rho})) - \gamma)) = \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_j)$ to highlight its dependence on the two samples $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$ and $\boldsymbol{z}_j = (\boldsymbol{x}_j, y_j)$. The proof contains the following six steps:

Step 1: In this step, we will bound the difference between $\mathcal{R}(\boldsymbol{W}_{\boldsymbol{z}}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W}_{\boldsymbol{z}})$ and its expectation $E_{\boldsymbol{z}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})]$, where $E_{\boldsymbol{z}}$ denotes the expectation with respect to the dataset $\boldsymbol{z}$. Let $\boldsymbol{z} = \{\boldsymbol{z}_1, ..., \boldsymbol{z}_{k-1}, \boldsymbol{z}_k, \boldsymbol{z}_{k+1}, ..., \boldsymbol{z}_n\}$ and $\boldsymbol{z}' = \{\boldsymbol{z}_1, ..., \boldsymbol{z}_{k-1}, \boldsymbol{z}'_k, \boldsymbol{z}_{k+1}, ..., \boldsymbol{z}_n\}$ be two sets of training samples that differ in one sample. Then we have the following inequality:

$$
\begin{aligned}
&|\sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})] - \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}'}(\boldsymbol{W}_{\boldsymbol{z}})]| \\
&\leq \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} |\mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}'}(\boldsymbol{W})| \\
&= \frac{\sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{j\neq k} |\Phi_{\boldsymbol{W}}(\boldsymbol{z}_k, \boldsymbol{z}_j) - \Phi_{\boldsymbol{W}}(\boldsymbol{z}'_k, \boldsymbol{z}_j)|}{n(n-1)} \\
&\leq \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \frac{2\sum_{j\neq k} |\Phi_{\boldsymbol{W}}(\boldsymbol{z}_k, \boldsymbol{z}_j)|}{n(n-1)} = \frac{2V_1}{n},
\end{aligned}
$$

where $|\Phi_{\boldsymbol{W}}(\boldsymbol{z}_k, \boldsymbol{z}_j)|$ is bounded by V_1 and $\boldsymbol{\mathcal{W}}$ is the weight space of the neural network. The above inequality shows that if we change the i-th sample $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$ while keeping all the others in $\boldsymbol{z}$ fixed, the value of the function $\sum_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})]$ does not change by more than $\frac{2V_1}{n}$. Applying McDiarmid's inequality [173] to the term $\sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})]$, with probability $1 - \delta$ there holds

$$
\Pr(|\sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})] - E_{\boldsymbol{z}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})]| > \epsilon) \leq \exp(-\frac{2\epsilon^2}{n * (2V_1/n)^2}), \tag{9.8}
$$

where ϵ is a constant. Based on the observation that $\mathcal{R}(\boldsymbol{W}_{\boldsymbol{z}}) - \mathcal{R}_z(\boldsymbol{W}_{\boldsymbol{z}}) \leq \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})]$, which allows us to go from working with $\mathcal{R}(\boldsymbol{W}_{\boldsymbol{z}}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W}_{\boldsymbol{z}})$ to working with $\sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})]$, the quantity $\mathcal{R}(\boldsymbol{W}_{\boldsymbol{z}}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W}_{\boldsymbol{z}})$ can be bounded as

$$
\mathcal{R}(\boldsymbol{W}_{\boldsymbol{z}}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W}_{\boldsymbol{z}}) \leq E_{\boldsymbol{z}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})] + V_1\sqrt{2\log(1/\delta)/n}. \tag{9.9}
$$

Step 2: Next, we will bound the expectation $E_{\boldsymbol{z}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})]$ in the right-hand side of Eqn. (9.9). Based on the Lemma 1 in [49], we can derive the inequality $E_{\boldsymbol{z}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W})] \leq E_{\boldsymbol{z}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \frac{1}{\lfloor\frac{n}{2}\rfloor}\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor} \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_{\lfloor\frac{n}{2}\rfloor+i})]$, where $\mathcal{R}_{\boldsymbol{z}} = \frac{2}{n(n-1)}\sum_{i<j} \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_j)$. Define $\hat{\mathcal{R}}_{\boldsymbol{z}}(\boldsymbol{W}) = \frac{1}{\lfloor\frac{n}{2}\rfloor}\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor} \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_{\lfloor\frac{n}{2}\rfloor+i})$. By introducing the ghost sample $\tilde{\boldsymbol{z}} = \{\tilde{\boldsymbol{z}}_1, \tilde{\boldsymbol{z}}_2, ..., \tilde{\boldsymbol{z}}_n\}$

where $\tilde{\boldsymbol{z}}_i$'s are independent of each other and of $\boldsymbol{z}_i$'s and have the same distribution as the latter, we have $E_{\boldsymbol{z}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\mathcal{R}(\boldsymbol{W}) - \hat{\mathcal{R}}_z(\boldsymbol{W})] = E_{\boldsymbol{z}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[E_{\tilde{\boldsymbol{z}}}[\hat{\mathcal{R}}_{\tilde{\boldsymbol{z}}}(\boldsymbol{W})] - \hat{\mathcal{R}}_{\boldsymbol{z}}(\boldsymbol{W})] \leq E_{\boldsymbol{z},\tilde{\boldsymbol{z}}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}}[\hat{\mathcal{R}}_{\tilde{\boldsymbol{z}}}(\boldsymbol{W}) - \hat{\mathcal{R}}_{\boldsymbol{z}}(\boldsymbol{W})]$, where $\mathcal{R}(\boldsymbol{W}) = E_{\boldsymbol{z}}[\hat{\mathcal{R}}_{\boldsymbol{z}}(\boldsymbol{W})] = E_{\tilde{\boldsymbol{z}}}[\hat{\mathcal{R}}_{\tilde{\boldsymbol{z}}}(\boldsymbol{W})]$ and the last inequality is based on Jensen's inequality. Then, we have

$$
\begin{aligned}
&E_{\boldsymbol{z},\tilde{\boldsymbol{z}}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} [\hat{\mathcal{R}}_{\tilde{\boldsymbol{z}}}(\boldsymbol{W}) - \hat{\mathcal{R}}_{\boldsymbol{z}}(\boldsymbol{W})] \\
&= E_{\boldsymbol{z},\tilde{\boldsymbol{z}}} \frac{1}{\lfloor \frac{n}{2} \rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} [\Phi_{\boldsymbol{W}}(\tilde{\boldsymbol{z}}_i, \tilde{\boldsymbol{z}}_{\lfloor \frac{n}{2} \rfloor + i}) - \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_{\lfloor \frac{n}{2} \rfloor + i})] \qquad (9.10) \\
&= E_{\boldsymbol{z},\tilde{\boldsymbol{z}},\boldsymbol{\xi}} \frac{1}{\lfloor \frac{n}{2} \rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i [\Phi_{\boldsymbol{W}}(\tilde{\boldsymbol{z}}_i, \tilde{\boldsymbol{z}}_{\lfloor \frac{n}{2} \rfloor + i}) - \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_{\lfloor \frac{n}{2} \rfloor + i})] \\
&\leq E_{\boldsymbol{z},\tilde{\boldsymbol{z}},\boldsymbol{\xi}} \frac{1}{\lfloor \frac{n}{2} \rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i \Phi_{\boldsymbol{W}}(\tilde{\boldsymbol{z}}_i, \tilde{\boldsymbol{z}}_{\lfloor \frac{n}{2} \rfloor + i}) \\
&\quad - E_{\boldsymbol{z},\tilde{\boldsymbol{z}},\boldsymbol{\xi}} \frac{1}{\lfloor \frac{n}{2} \rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_{\lfloor \frac{n}{2} \rfloor + i}) = 2E_{\boldsymbol{z},\boldsymbol{\xi}} \frac{1}{\lfloor \frac{n}{2} \rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_{\lfloor \frac{n}{2} \rfloor + i}).
\end{aligned}
$$

In the second equality of Eqn. (9.10), we introduce the Rademacher random variables $\{\xi_i\}_{i=1}^{\lfloor \frac{n}{2} \rfloor}$ [50], and use the fact that multiplying $[\Phi_{\boldsymbol{W}}(\tilde{\boldsymbol{z}}_i, \tilde{\boldsymbol{z}}_{\lfloor \frac{n}{2} \rfloor + i}) - \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_{\lfloor \frac{n}{2} \rfloor + i})]$ by a Rademacher variable ξ_i does not change the expectation of the sum [50]. The first inequality of Eqn. (9.10) follows from the sub-additivity of the supremum function. For the last equality in Eqn. (9.10), since $\boldsymbol{z}$ and $\tilde{\boldsymbol{z}}$ have the same distribution, we change the sign to simplify the expression. Then, Eqn. (9.9) can be rewritten as

$$
\mathcal{R}(\boldsymbol{W}_{\boldsymbol{z}}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W}_{\boldsymbol{z}}) \leq 2E_{\boldsymbol{z},\boldsymbol{\xi}} \frac{1}{\lfloor \frac{n}{2} \rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_{\lfloor \frac{n}{2} \rfloor + i}) + V_1 \sqrt{2\log(1/\delta)/n}. \quad (9.11)
$$

Step 3: Then we bound $2E_{\boldsymbol{z},\boldsymbol{\xi}} \frac{1}{\lfloor \frac{n}{2} \rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_{\lfloor \frac{n}{2} \rfloor + i})$ in Eqn. (9.11) as follows

$$
\begin{aligned}
&2E_{\boldsymbol{z},\boldsymbol{\xi}} \frac{1}{\lfloor \frac{n}{2} \rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i \Phi_{\boldsymbol{W}}(\boldsymbol{z}_i, \boldsymbol{z}_{\lfloor \frac{n}{2} \rfloor + i}) \leq 6\sqrt{1/\lfloor \frac{n}{2} \rfloor} \\
&+ 2V_1 \sqrt{2\log(1/\delta)/n} + \sum_{k=1}^{h^L} 4E_{\boldsymbol{\xi}} \frac{1}{\lfloor \frac{n}{2} \rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i (f_k^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho}) - f_k^L(\boldsymbol{x}_{\lfloor \frac{n}{2} \rfloor + i}, \boldsymbol{W}, \boldsymbol{\rho}))^2,
\end{aligned}
$$

where $f_k^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho}) = E_{\boldsymbol{M}}[(\boldsymbol{W}_k^L \odot \boldsymbol{M}_k^L)\sigma(f^{L-1}(\boldsymbol{x}_i, \boldsymbol{W}^{1:L-1}, \boldsymbol{M}^{1:L-1}))]$ and $\boldsymbol{W}_k^L$ denotes the k-th row of $\boldsymbol{W}^L$.

Step 4: For the term $\sum_{k=1}^{h^L} 4E_{\boldsymbol{\xi}} \frac{1}{\lfloor \frac{n}{2} \rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i (f_k^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho}) - f_k^L(\boldsymbol{x}_{\lfloor \frac{n}{2} \rfloor + i}, \boldsymbol{W}, \boldsymbol{\rho}))^2$ in the above equation, we here aim to establish its upper bound. Based on the assumption $\|\boldsymbol{W}^L\|_F^2 = \sum_{k=1}^{h^L} \|\boldsymbol{W}_k^L\|^2 \leq (B^L)^2$, we can derive that $\forall k \in [h^L]$, $\|\boldsymbol{W}_k^L\|^2 \leq (B^L)^2$. Based on the Cauchy-Schwarz inequality, we have $|f_k^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho})| \leq \sup B^L \|\sigma(f^{L-1}(\cdot))\|$. Assume that $\|\sigma(f^{L-1}(\cdot))\|$ is bounded in $[0, V^L]$. Then we can have that $(f_k^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho}) - f_k^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho})) \in [-2B^L V^L, 2B^L V^L]$.

Define the function $\phi(t) = t^2 (t \in [-2B^L V^L, 2B^L V^L])$ with $\phi(0) = 0$. Then, we have that the defined function $\phi(t) = t^2 (t \in [-2B^L V^L, 2B^L V^L])$ is Lipschitz continuous with Lipschitz constant $4B^L V^L$. Then, the first term in the right-hand side of the above equation can be bounded as

$$
\begin{aligned}
&\sum_{k=1}^{h^L} 4E_{\boldsymbol{\xi}} \frac{1}{\lfloor \frac{n}{2} \rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i (f_k^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho}) - f_k^L(\boldsymbol{x}_{\lfloor \frac{n}{2} \rfloor + i}, \boldsymbol{W}, \boldsymbol{\rho}))^2 \\
&\leq \frac{16 h^L B^L V^L}{\lfloor \frac{n}{2} \rfloor} [E_{\boldsymbol{\xi}} \sup_{k\in[h^L]} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} |\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i f_k^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho})| \\
&+ E_{\boldsymbol{\xi}} \sup_{k\in[h^L]} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} |\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i f_k^L(\boldsymbol{x}_{\lfloor \frac{n}{2} \rfloor + i}, \boldsymbol{W}, \boldsymbol{\rho})|],
\end{aligned}
$$

where $f_k^L(\boldsymbol{x}_{\lfloor \frac{n}{2} \rfloor + i}, \boldsymbol{W}, \boldsymbol{\rho})$ is the deterministic output on the L-th layer.

Step 5: In this step, for the two terms $E_{\boldsymbol{\xi}} \sup_{k\in[h^L]} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} |\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i f_k^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho})|$ and $E_{\boldsymbol{\xi}} \sup_{k\in[h^L]} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} |\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i f_k^L(x_{\lfloor \frac{n}{2} \rfloor + i}, \boldsymbol{W}, \boldsymbol{\rho})|$ in the right-hand side of the above equation, we will bound them in a recursive way. With a previously specified constant $\lambda > 0$, the first term $E_{\boldsymbol{\xi}} \sup_{k\in[h^L]} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} |\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i f_k^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho})|$ can be bounded as

$$
\begin{aligned}
&E_{\boldsymbol{\xi}} \sup_{k\in[h^L]} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} |\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i f_k^L(\boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{\rho})| \\
&\leq \frac{1}{\lambda} \log[2^L * E_{\boldsymbol{\xi}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \exp((\prod_{l=1}^{L} B^l) * \prod_{l=1}^{L} \sqrt{\rho^l} * \lambda * \|\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i \boldsymbol{x}_i\|)].
\end{aligned}
$$

Similarly, the second term $E_{\boldsymbol{\xi}} \sup_{k\in[h^L]} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} |\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \xi_i f_k^L(\boldsymbol{x}_{i+\lfloor \frac{n}{2} \rfloor})|$ can be bounded as

$\frac{1}{\lambda}\log[2^L * E_{\boldsymbol{\xi}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \exp((\prod_{l=1}^{L} B^l) * \prod_{l=1}^{L}\sqrt{\rho^l} * \lambda * \|\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\xi_i \boldsymbol{x}_{i+\lfloor\frac{n}{2}\rfloor}\|)]$. Thus, we have

$$
\begin{aligned}
&\mathcal{R}(\boldsymbol{W}_{\boldsymbol{z}}) - \mathcal{R}_{\boldsymbol{z}}(\boldsymbol{W}_{\boldsymbol{z}}) \\
&\leq 6\sqrt{\frac{1}{\lfloor\frac{n}{2}\rfloor}} + \frac{16h^L B^L V^L}{\lambda\lfloor\frac{n}{2}\rfloor}[\log[2^L * E_{\boldsymbol{\xi}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \exp(\lambda \prod_{l=1}^{L}(B^l\sqrt{\rho^l}) * \|\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\xi_i \boldsymbol{x}_i\|)] \\
&+ \log[2^L * E_{\boldsymbol{\xi}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \exp(\lambda \prod_{l=1}^{L}(B^l\sqrt{\rho^l}) * \|\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\xi_i \boldsymbol{x}_{i+\lfloor\frac{n}{2}\rfloor}\|)]] + 3V_1\sqrt{\frac{2\log(1/\delta)}{n}}.
\end{aligned}
$$

Step 6: Here, we aim to address how to estimate the two terms in the above equations (i.e., $(\log[2^L * E_{\boldsymbol{\xi}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \exp((\prod_{l=1}^{L} B^l) * \prod_{l=1}^{L}\sqrt{\rho^l} * \lambda * \|\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\xi_i \boldsymbol{x}_i\|)])/\lambda$ and $(\log[2^L * E_{\boldsymbol{\xi}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \exp((\prod_{l=1}^{L} B^l) * \prod_{l=1}^{L}\sqrt{\rho^l} * \lambda * \|\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\xi_i \boldsymbol{x}_{i+\lfloor\frac{n}{2}\rfloor}\|)])/\lambda$ in the right-hand side of Eqn. (9.12)). Define a random variable $Z_1 = \prod_{l=1}^{L} B^l * \prod_{l=1}^{L}\sqrt{\rho^l} * \|\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\xi_i \boldsymbol{x}_i\|$, and then the first term can be decomposed as

$$
\begin{aligned}
&(\log[2^L E_{\boldsymbol{\xi}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \exp((\prod_{l=1}^{L} B^l) \prod_{l=1}^{L}\sqrt{\rho^l}\lambda\|\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\xi_i \boldsymbol{x}_i\|)])/\lambda = \\
&\qquad (L\log 2)/\lambda + (\log[E_{\boldsymbol{\xi}} \exp\lambda(Z_1 - EZ_1)])/\lambda + EZ_1,
\end{aligned}
$$

where the first term $(L\log 2)/\lambda$ in the right-hand side is a constant. The second and the third terms in the right-hand side of Eqn. (9.12) can be upper bounded as $\log[E_{\boldsymbol{\xi}} \exp\lambda(Z_1 - EZ_1)]/\lambda \leq \lambda(\prod_{l=1}^{L-1} B^l\sqrt{\rho^l})^2 \sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\|\boldsymbol{x}_i\|^2/2$ and $EZ_1 \leq \prod_{l=1}^{L}(B^l\sqrt{\rho^l})\sqrt{\|\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\boldsymbol{x}_i\|^2}$, respectively.

Next, by setting $\lambda = \frac{\sqrt{2\log(2)L}}{\prod_{l=1}^{L}(B^l\sqrt{\rho^l})\sqrt{\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\|\boldsymbol{x}_i\|^2}}$, the second term in the right-hand side of Eqn. (9.12) can be bounded as $\log[2^L * E_{\boldsymbol{\xi}} \frac{1}{\lfloor\frac{n}{2}\rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \exp(\lambda \prod_{l=1}^{L}(B^l\sqrt{\rho^l})\|\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\xi_i \boldsymbol{x}_i\|)]/\lambda \leq (\sqrt{2L\log 2}+1)\prod_{l=1}^{L}(B^l\sqrt{\rho^l})\sqrt{\lfloor\frac{n}{2}\rfloor d}$. In a similar way, we can bound the third term $(\log[2^L * E_{\boldsymbol{\xi}} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \exp((\prod_{l=1}^{L} B^l) * \prod_{l=1}^{L}\sqrt{\rho^l} * \lambda * \|\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\xi_i \boldsymbol{x}_{i+\lfloor\frac{n}{2}\rfloor}\|)])/\lambda$ in the right-hand side of Eqn. (9.12) as

$$
\begin{aligned}
&(\log[2^L * E_{\boldsymbol{\xi}} \frac{1}{\lfloor\frac{n}{2}\rfloor} \sup_{\boldsymbol{W}\in\boldsymbol{\mathcal{W}}} \exp(\lambda(\prod_{l=1}^{L} B^l) * \prod_{l=1}^{L}\sqrt{\rho^l} * \|\sum_{i=1}^{\lfloor\frac{n}{2}\rfloor}\xi_i \boldsymbol{x}_{i+\lfloor\frac{n}{2}\rfloor}\|)])/\lambda \leq \\
&\qquad (\sqrt{2L\log 2}+1)\prod_{l=1}^{L}(B^l\sqrt{\rho^l})\sqrt{\lfloor\frac{n}{2}\rfloor d}.
\end{aligned}
$$

Combining the previous results, we complete the proof. □

9.4 Proof of Theorem 4

Before presenting the proof of Theorem 4, we firstly introduce the assumptions, the definition and the concept of the attacked MDPs, which are used in the latter proof.

Assumption 3 (MDP Regularity)**.** *Suppose MDP $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ has finite state space, finite action space, and bounded reward function, i.e., $|\mathcal{S}| \leq \infty$, $|\mathcal{A}| \leq \infty$, and $\|\mathcal{R}\|_\infty \leq R$, where R is the upper bound of reward function.*

Assumption 4 (Transition/Reward Continuity)**.** *Suppose two MDPs: $\mathcal{M}' =< \mathcal{S}, \mathcal{A}, \mathcal{P}', \mathcal{R}', \gamma >$ and $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ share the same state space and action space. $\mathcal{P}(\cdot|\boldsymbol{s}, a) = \mathcal{P}'(\cdot|\tilde{\boldsymbol{s}}, a)$ and $\|\boldsymbol{s} - \tilde{\boldsymbol{s}}\| \leq \epsilon$. Suppose there exists a constant δ such that for any $\boldsymbol{s} \in \mathcal{S}$ and $a \in \mathcal{A}$, $\|\mathcal{P}'(\cdot|\boldsymbol{s}, a) - \mathcal{P}(\cdot|\boldsymbol{s}, a)\|_1 \leq \delta$ and $|\mathcal{R}'(\boldsymbol{s}, a) - \mathcal{R}(\boldsymbol{s}, a)| \leq \delta$. In some literature, $\mathcal{M}$ is also said to be δ-approximation of $\mathcal{M}'$.*

Definition 7 ($\mathcal{T}$-step value function)**.** *Suppose MDP $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ satisfies Assumption 1. Following the definition of value function under policy π: $V^\pi(\boldsymbol{s}) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(\boldsymbol{s}_t, a_t = \pi(\boldsymbol{s}_t))|\boldsymbol{s}_0 = \boldsymbol{s}]$ in previous section, we define $\mathcal{T}$-step value function to be the truncation of the first $\mathcal{T}$ discounted returns. Specifically, $\mathcal{V}^\pi(\boldsymbol{s}, \mathcal{T}) = \mathbb{E}[\sum_{t=0}^{\mathcal{T}-1} \gamma^t \mathcal{R}(\boldsymbol{s}_t, \pi(\boldsymbol{s}_t))|\boldsymbol{s}_0 = \boldsymbol{s}]$. Note the expectation is taken over all possible paths the agent might follow starting from $\boldsymbol{s}$ and of fixed length $\mathcal{T}$. Further, we define the optimal $\mathcal{T}$-step value function $\mathcal{V}^*(\boldsymbol{s}, \mathcal{T}) = \max_\pi \mathcal{V}^\pi(\boldsymbol{s}, \mathcal{T})$. The optimal value function could be viewed as the limit case of optimal $\mathcal{T}$-step value function, i.e., $\mathcal{V}^*(\boldsymbol{s}) = \lim_{\mathcal{T}\to\infty} \mathcal{V}^*(\boldsymbol{s}, \mathcal{T})$. Finally, we denote the maximum possible $\mathcal{T}$-step return by $\mathcal{G}_\mathcal{T} = \max_{\boldsymbol{s}\in\mathcal{S}} \mathcal{V}^*(\boldsymbol{s}, \mathcal{T})$.*

Definition 8 (The attacked MDPs)**.** *Here, we define the following two MDPs, which are different from original MDP $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ attempted to be attacked in either transition probability or immediate reward, indicating the new environments under malicious attacks.*

- *$\mathcal{M}_1 =< \mathcal{S}, \mathcal{A}, \mathcal{P}_1, \mathcal{R}_1, \gamma >$, where $\mathcal{P}_1(\cdot|\boldsymbol{s}, a) = \mathcal{P}(\cdot|\tilde{\boldsymbol{s}}, a)$, and $\mathcal{R}_1(\cdot|\boldsymbol{s}, a) = \mathcal{R}(\cdot|\boldsymbol{s}, a)$. $\mathcal{P}_1$ characterizes the system dynamics where the next state follows the distribution $\mathcal{P}(\cdot|\tilde{\boldsymbol{s}}, a)$ given the current state is $\boldsymbol{s}$ and the selected action is a, since the current state has been crafted to $\tilde{\boldsymbol{s}}$.*
- *$\mathcal{M}_2 =< \mathcal{S}, \mathcal{A}, \mathcal{P}_2, \mathcal{R}_2, \gamma >$, where $\mathcal{P}_2(\cdot|\boldsymbol{s}, a) = \mathcal{P}(\cdot|\tilde{\boldsymbol{s}}, a)$, and $\mathcal{R}_2(\boldsymbol{s}, a) = \mathcal{R}(\tilde{\boldsymbol{s}}, a)$. The system transition model $\mathcal{P}_2$ is exactly the same as $\mathcal{P}_1$, while the immediate reward model $\mathcal{R}_2$ is different. The reward the agent instantaneously collects from the environment is*

$\mathcal{R}(\tilde{\boldsymbol{s}}, a)$ in $\mathcal{M}_2$ given that the current state is $\boldsymbol{s}$ and the selected action is a. We argue that both $\mathcal{R}_2$ and $\mathcal{R}_1$ are ubiquitous in real-world settings, depending on whether the environment from which the agent collects reward is aware of the state crafting. $\mathcal{M}_1$ characterizes the scenario where the attack is imperceptible to the system or the agent obtains reward based on his own assessment of the current state and action. On the contrary, $\mathcal{M}_2$ describes the setting where the immediate reward is evaluated externally and the attack is noticeable to the performance evaluation system.

Finding optimal malicious attack is equivalent to solving the optimal policy for $\mathcal{M}_1$ and $\mathcal{M}_2$. In Theorem 4, we bound the difference of optimal value functions between $\mathcal{M}_i$ and $\mathcal{M}$, where $i = 1, 2$. The importance of these bounds is to help us understand, to what extent the malicious attack affects the long term reward and how perceptible the attack could be to the participating agent. In Theorem 5, we further study, if the attack takes place only in a finite number of states, the difference of $\mathcal{T}$-step value functions between $\mathcal{M}_i$ and $\mathcal{M}$, where $i = 1, 2$.

After introducing the assumptions, the definition and the concept of the attacked MDPs, we next present the proof of Theorem 4.

Proof. Recall that Bellman's Equation for the optimal value function is as follows: $V^*(\boldsymbol{s}) = \max_{a\in\mathcal{A}} \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}(\cdot|\boldsymbol{s},a)} [\mathcal{R}(\boldsymbol{s}, a) + \gamma V^*(\boldsymbol{s}')]$. Let $\mathcal{F}$ be the function class mapping from $\mathcal{S}$ to $\mathbb{R}$. Define Bellman's operator $\mathcal{B} : \mathcal{F} \to \mathcal{F}$ of MDP $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ to be $\mathcal{B} \circ V^*(\boldsymbol{s}) = \max_{a\in\mathcal{A}} \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}(\cdot|\boldsymbol{s},a)} [\mathcal{R}(\boldsymbol{s}, a) + \gamma V^*(\boldsymbol{s}')]$.

$$
\begin{aligned}
&|V^*(\boldsymbol{s}) - V_1^*(\boldsymbol{s})| = |\mathcal{B} \circ V^*(\boldsymbol{s}) - \mathcal{B} \circ V_1^*(\boldsymbol{s})| \\
&= \max\{\mathcal{B} \circ V^*(\boldsymbol{s}) - \mathcal{B} \circ V_1^*(\boldsymbol{s}), \mathcal{B} \circ V_1^*(\boldsymbol{s}) - \mathcal{B} \circ V^*(\boldsymbol{s})\} \\
&\underset{\mathbf{1}}{\leq} \max_{a\in\mathcal{A}} |\mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}(\cdot|\boldsymbol{s},a)}[\mathcal{R}(\boldsymbol{s}, a) + \gamma V^*(\boldsymbol{s}')] - \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}_1(\cdot|\boldsymbol{s},a)}[\mathcal{R}_1(\boldsymbol{s}, a) + \gamma V_1^*(\boldsymbol{s}')]|.
\end{aligned}
$$

Here, for $\hat{a}$ and $\hat{a}_1$, we suppose that $\hat{a} \triangleq \arg\max_{a\in\mathcal{A}} \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}(\cdot|\boldsymbol{s},a)} [\mathcal{R}(\boldsymbol{s}, a) + \gamma V^*(\boldsymbol{s}')]$, and $\hat{a}_1 \triangleq \arg\max_{a\in\mathcal{A}} \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}_1(\cdot|\boldsymbol{s},a)} [\mathcal{R}_1(\boldsymbol{s}, a) + \gamma V_1^*(\boldsymbol{s}')]$. The reason for inequality **1** is: if $V^*(\boldsymbol{s}) \geq V_1^*(\boldsymbol{s})$, we can derive that $\mathcal{B} \circ V^*(\boldsymbol{s}) - \mathcal{B} \circ V_1^*(\boldsymbol{s}) \leq \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}(\cdot|\boldsymbol{s},\hat{a})} [\mathcal{R}(\boldsymbol{s}, \hat{a}) + \gamma V^*(\boldsymbol{s}')] - \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}_1(\cdot|\boldsymbol{s},\hat{a})} [\mathcal{R}_1(\boldsymbol{s}, \hat{a}) + \gamma V_1^*(\boldsymbol{s}')]$; while if $V^*(\boldsymbol{s}) < V_1^*(\boldsymbol{s})$, we can then obtain that $\mathcal{B} \circ V_1^*(\boldsymbol{s}) - \mathcal{B} \circ V^*(\boldsymbol{s}) \leq \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}_1(\cdot|\boldsymbol{s},\hat{a}_1)} [\mathcal{R}_1(\boldsymbol{s}, \hat{a}_1) + \gamma V_1^*(\boldsymbol{s}')] - \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}(\cdot|\boldsymbol{s},\hat{a}_1)} [\mathcal{R}(\boldsymbol{s}, \hat{a}_1) + \gamma V^*(\boldsymbol{s}')]$.

In both cases, the inequality **1** holds.

$$
\begin{aligned}
&|V^*(\boldsymbol{s}) - V_1^*(\boldsymbol{s})| \leq \max_{a\in\mathcal{A}} |\mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}(\cdot|\boldsymbol{s},a)}[\gamma V^*(\boldsymbol{s}')] - \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}_1(\cdot|\boldsymbol{s},a)}[\gamma V_1^*(\boldsymbol{s}')]| \\
&\underset{\mathbf{2}}{\leq} \underbrace{\max_{a\in\mathcal{A}} |\mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}(\cdot|\boldsymbol{s},a)}[\gamma V^*(\boldsymbol{s}')] - \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}_1(\cdot|\boldsymbol{s},a)}[\gamma V^*(\boldsymbol{s}')]|}_{I} \\
&+ \underbrace{\max_{a\in\mathcal{A}} |\mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}_1(\cdot|\boldsymbol{s},a)}[\gamma V^*(\boldsymbol{s}')] - \mathbb{E}_{\boldsymbol{s}'\sim\mathcal{P}_1(\cdot|\boldsymbol{s},a)}[\gamma V_1^*(\boldsymbol{s}')]|}_{II}.
\end{aligned}
$$

Inequality **2** follows from the triangle inequality. It is easy to get $II \leq \gamma \|V^* - V_1^*\|_\infty$. We now derive the bound for I.

$$
\begin{aligned}
I &= \max_{a\in\mathcal{A}} |\sum_{\boldsymbol{s}'\in\mathcal{S}} (\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},a) - \mathcal{P}_1(\boldsymbol{s}'|\boldsymbol{s},a)) V^*(\boldsymbol{s}')| \cdot \gamma \\
&\underset{\mathbf{3}}{\leq} \max_{a\in\mathcal{A}} \|\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},a) - \mathcal{P}_1(\boldsymbol{s}'|\boldsymbol{s},a)\|_1 \max_{\boldsymbol{s}\in\mathcal{S}} V^*(\boldsymbol{s}) \cdot \gamma \\
&\underset{\mathbf{4}}{\leq} \max_{a\in\mathcal{A}} 2 \cdot TV(\mathcal{P}(\cdot|\boldsymbol{s},a), \mathcal{P}_1(\cdot|\boldsymbol{s},a)) \cdot \max_{\boldsymbol{s}\in\mathcal{S}} V^*(\boldsymbol{s}) \cdot \gamma \\
&\underset{\mathbf{5}}{\leq} \max_{a\in\mathcal{A}} 2 \cdot TV(\mathcal{P}(\cdot|\boldsymbol{s},a), \mathcal{P}_1(\cdot|\boldsymbol{s},a)) \cdot \frac{R}{1-\gamma} \cdot \gamma.
\end{aligned}
$$

Inequality **3** follows from Hölder's inequality. Let $\mathcal{S}_0 \triangleq \{\boldsymbol{s}' \in S : \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},a) \geq \mathcal{P}_1(\boldsymbol{s}'|\boldsymbol{s},a)\}$. We have $\|\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},a) - \mathcal{P}_1(\boldsymbol{s}'|\boldsymbol{s},a)\|_1 = \sum_{\boldsymbol{s}'\in\mathcal{S}_0} (\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},a) - \mathcal{P}_1(\boldsymbol{s}'|\boldsymbol{s},a)) + \sum_{\boldsymbol{s}'\in\mathcal{S}\setminus\mathcal{S}_0} (\mathcal{P}_1(\boldsymbol{s}'|\boldsymbol{s},a) - \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},a)) = 2\mathcal{P}(\mathcal{S}_0|\boldsymbol{s},a) - 2\mathcal{P}_1(\mathcal{S}_0|\boldsymbol{s},a) \leq 2 \cdot TV(\mathcal{P}(\cdot|\boldsymbol{s},a), \mathcal{P}_1(\cdot|\boldsymbol{s},a))$. Therefore, inequality **4** holds. Recall $V(\boldsymbol{s}) \triangleq \mathbb{E}[\sum_{t\geq 0} \gamma^t \mathcal{R}(\boldsymbol{s}_t, a_t)] \leq \mathbb{E}[\sum_{t\geq 0} \gamma^t R] \leq \frac{R}{1-\gamma}$. Hence, inequality **5** holds. With all these combined, we have the following

$$
\begin{aligned}
&\|V^* - V_1^*\|_\infty = \max_{\boldsymbol{s}\in\mathcal{S}} |V^*(\boldsymbol{s}) - V_1^*(\boldsymbol{s})| \\
&\leq \max_{a\in\mathcal{A}, \boldsymbol{s}\in\mathcal{S}} 2 \cdot TV(\mathcal{P}(\cdot|\boldsymbol{s},a), \mathcal{P}_1(\cdot|\boldsymbol{s},a)) \cdot \frac{R}{1-\gamma} \cdot \gamma + \|V^* - V_1^*\|_\infty \cdot \gamma.
\end{aligned}
$$

Based on the above, by reorganizing the last inequality, we can have that $\|V^* - V_1^*\|_\infty \leq \frac{2\gamma R}{(1-\gamma)^2} \max_{\boldsymbol{s}\in\mathcal{S}, a\in\mathcal{A}} TV(\mathcal{P}(\cdot|\boldsymbol{s},a), \mathcal{P}_1(\cdot|\boldsymbol{s},a))$. Recall that $\mathcal{P}$ and $\mathcal{P}_1$ are $L\epsilon$ approximate. Therefore, together with inequality 3, we have

$$
\begin{aligned}
I &\underset{\mathbf{3}}{\leq} \max_{a\in\mathcal{A}} \|\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},a) - \mathcal{P}_1(\boldsymbol{s}'|\boldsymbol{s},a)\|_1 \max_{\boldsymbol{s}\in\mathcal{S}} V^*(\boldsymbol{s}) \cdot \gamma \\
&\leq L\epsilon \cdot \frac{R}{1-\gamma} \cdot \gamma.
\end{aligned}
$$

With II combined, we have $\|V^* - V_1^*\|_\infty \leq \frac{\gamma RL}{(1-\gamma)^2}\epsilon$, which completes the first part of proof.

$$
\begin{aligned}
&|V^*(\boldsymbol{s}) - V_2^*(\boldsymbol{s})| = |\mathcal{B} \circ V^*(\boldsymbol{s}) - \mathcal{B} \circ V_2^*(\boldsymbol{s})| \\
&\underset{\mathbf{6}}{\leq} \max_{a \in \mathcal{A}} |\mathbb{E}_{\boldsymbol{s}' \sim \mathcal{P}(\cdot|\boldsymbol{s},a)}[\gamma V^*(\boldsymbol{s}')] - \mathbb{E}_{\boldsymbol{s}' \sim \mathcal{P}_2(\cdot|\boldsymbol{s},a)}[\gamma V_2^*(\boldsymbol{s}')]| \\
&\qquad + \max_{a \in \mathcal{A}} |\mathcal{R}(\boldsymbol{s}, a) - \mathcal{R}_2(\boldsymbol{s}, a)| \\
&\leq \max_{a \in \mathcal{A}} |\mathbb{E}_{\boldsymbol{s}' \sim \mathcal{P}(\cdot|\boldsymbol{s},a)}[\gamma V^*(\boldsymbol{s}')] - \mathbb{E}_{\boldsymbol{s}' \sim \mathcal{P}_2(\cdot|\boldsymbol{s},a)}[\gamma V_2^*(\boldsymbol{s}')]| + l\epsilon.
\end{aligned}
$$

Suppose $\hat{a} \triangleq \arg\max_{a \in \mathcal{A}} \mathcal{R}(\boldsymbol{s}, a)$, and $\hat{a}_2 \triangleq \arg\max_{a \in \mathcal{A}} \mathcal{R}_2(\boldsymbol{s}, a)$. If $\max_{a \in \mathcal{A}} \mathcal{R}(\boldsymbol{s}, a) \geq \max_{a \in \mathcal{A}} \mathcal{R}_2(\boldsymbol{s}, a)$, $|\max_{a \in \mathcal{A}} \mathcal{R}(\boldsymbol{s}, a) - \max_{a \in \mathcal{A}} \mathcal{R}_2(\boldsymbol{s}, a)| \leq \mathcal{R}(\boldsymbol{s}, \hat{a}) - \mathcal{R}_2(\boldsymbol{s}, \hat{a})$; otherwise, $|\max_{a \in \mathcal{A}} \mathcal{R}(\boldsymbol{s}, a) - \max_{a \in \mathcal{A}} \mathcal{R}_2(\boldsymbol{s}, a)| \leq \mathcal{R}(\boldsymbol{s}, \hat{a}_2) - \mathcal{R}_2(\boldsymbol{s}, \hat{a}_2)$. In both cases, $|\max_{a \in \mathcal{A}} \mathcal{R}(\boldsymbol{s}, a) - \max_{a \in \mathcal{A}} \mathcal{R}_2(\boldsymbol{s}, a)| \leq \max_{a \in \mathcal{A}} |\mathcal{R}(\boldsymbol{s}, a) - \mathcal{R}_2(\boldsymbol{s}, a)|$. Together with inequality 1 and triangle inequality, inequality 6 holds. The rest is the same as the first part of theorem. □

9.5 Proof of Theorem 5

Proof. Let ψ be any path of states starting from $\boldsymbol{s}$ under policy π^* of length $\mathcal{T}$, i.e., $\psi = (\boldsymbol{s}_0^\psi = \boldsymbol{s}, \boldsymbol{s}_1^\psi, \boldsymbol{s}_2^\psi, ..., \boldsymbol{s}_{\mathcal{T}-1}^\psi)$. Let Ψ be the set of all possible such kind of paths. Further, we define $V^{\pi^*}(\psi) = \sum_{t=0}^{\mathcal{T}-1} \gamma^t \mathcal{R}(\boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi))$, and $\mathbb{P}^{\pi^*}[\psi] = \prod_{t=0}^{\mathcal{T}-2} \mathcal{P}(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi))$. $V^{\pi^*}(\psi)$ is the total discounted reward the agent could receive along path ψ under policy π^* in $\mathcal{M}$, and $\mathbb{P}^{\pi^*}[\psi]$ is the probability of the agent taking path ψ under policy π^* in $\mathcal{M}$. Analogously, we have $V_i^{\pi^*}(\psi)$ and $\mathbb{P}_i^{\pi^*}[\psi]$.

The proof is a modification of Lemma 4 in [174]. According to the definition of $\mathcal{T}$-step value function under policy π^*, we know $\mathcal{V}^{\pi^*}(\boldsymbol{s}, \mathcal{T}) = \sum_{\psi \in \Psi} \mathbb{P}^{\pi^*}[\psi] V^{\pi^*}(\psi)$. We define the set of all θ-small paths, $\Psi_1 = \{\psi \in \Psi : \exists t, \mathcal{P}(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi)) \leq \theta\}$. Naturally, the set of paths where all transition probabilities under π^* is larger than θ is $\Psi_2 = \Psi \setminus \Psi_1$. Thus, we have $\mathcal{V}^{\pi^*}(\boldsymbol{s}, \mathcal{T}) = \sum_{\psi \in \Psi_1} \mathbb{P}^{\pi^*}[\psi] V^{\pi^*}(\psi) + \sum_{\psi \in \Psi_2} \mathbb{P}^{\pi^*}[\psi] V^{\pi^*}(\psi)$.

We have $|\mathcal{P}_i(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi)) - \mathcal{P}(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi))| \leq \delta$, following from the continuity assumption. For any path $\psi \in \Psi_2$, for all $0 \leq t \leq \mathcal{T} - 2$, it is not difficult to see $\mathcal{P}(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi)) + \frac{\delta}{\theta}\theta \leq \mathcal{P}(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi)) + \frac{\delta}{\theta}\mathcal{P}(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi))$, and $\mathcal{P}(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi)) - \frac{\delta}{\theta}\theta \geq \mathcal{P}(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi)) - \frac{\delta}{\theta}\mathcal{P}(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi))$. Therefore, we have

$$
\begin{aligned}
&(1 - \frac{\delta}{\theta})\mathcal{P}(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi)) \leq \mathcal{P}_i(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi)) \\
&\leq (1 + \frac{\delta}{\theta})\mathcal{P}(\boldsymbol{s}_{t+1}^\psi | \boldsymbol{s}_t^\psi, \pi^*(\boldsymbol{s}_t^\psi)).
\end{aligned}
$$

Furthermore, we can derive that for any $\psi \in \Psi_2$, $(1-\frac{\delta}{\theta})^{\mathcal{T}}\mathbb{P}^{\pi^*}[\psi] \leq \mathbb{P}_i^{\pi^*}[\psi] \leq (1+\frac{\delta}{\theta})^{\mathcal{T}}\mathbb{P}^{\pi^*}[\psi]$.

Now let us study the property of path $\psi \in \Psi_1$. $\sum_{\psi\in\Psi_1}\mathbb{P}_i^{\pi^*}[\psi]V^{\pi^*}(\psi)$ is upper bounded by $\theta\cdot|\mathcal{S}|\cdot\mathcal{T}\mathcal{G}_{\mathcal{T}}$ and $\sum_{\psi\in\Psi_1}\mathbb{P}^{\pi^*}[\psi]V_i^{\pi^*}(\psi)$ is upper bounded by $(\theta+\delta)\cdot|\mathcal{S}|\cdot\mathcal{T}\mathcal{G}_{\mathcal{T}}$. Thus, we have $|\sum_{\psi\in\Psi_1}\mathbb{P}_i^{\pi^*}[\psi]V_i^{\pi^*}(\psi) - \sum_{\psi\in\Psi_1}\mathbb{P}^{\pi^*}[\psi]V^{\pi^*}(\psi)| \leq (\delta+2\theta)|\mathcal{S}|\mathcal{T}\mathcal{G}_{\mathcal{T}}$.
For any path ψ, $|V^{\pi^*}(\psi) - V_i^{\pi^*}(\psi)| \leq \mathcal{T}\delta$. With Ψ_1 and Ψ_2 combined, we have

$$
\begin{aligned}
&\mathcal{V}_i^{\pi^*}(\boldsymbol{s},\mathcal{T}) \triangleq \sum_{\psi\in\Psi_1}\mathbb{P}_i^{\pi^*}[\psi]V_i^{\pi^*}(\psi) + \sum_{\psi\in\Psi_2}\mathbb{P}_i^{\pi^*}[\psi]V_i^{\pi^*}(\psi)\\
&\leq \{\sum_{\psi\in\Psi_1}\mathbb{P}^{\pi^*}[\psi]V^{\pi^*}(\psi) + (\delta+2\theta)|\mathcal{S}|\mathcal{T}\mathcal{G}_{\mathcal{T}}\} + \sum_{\psi\in\Psi_2}\mathbb{P}_i^{\pi^*}[\psi]V_i^{\pi^*}(\psi)\\
&\leq \{V_{\Psi_1}^{\pi^*}(\boldsymbol{s},\mathcal{T}) + (\delta+2\theta)|\mathcal{S}|\mathcal{T}\mathcal{G}_{\mathcal{T}}\} + \sum_{\psi\in\Psi_2}(1+\frac{\delta}{\theta})^{\mathcal{T}}\mathbb{P}^{\pi^*}[\psi](V^{\pi^*}(\psi)+\mathcal{T}\delta)\\
&\leq (1+\frac{\delta}{\theta})^{\mathcal{T}}(V_{\Psi_1}^{\pi^*}(\boldsymbol{s},\mathcal{T}) + V_{\Psi_2}^{\pi^*} + \mathcal{T}\delta) + (\delta+2\theta)|\mathcal{S}|\mathcal{T}\mathcal{G}_{\mathcal{T}}\\
&\leq (1+\frac{\delta}{\theta})^{\mathcal{T}}(\mathcal{V}^{\pi^*}(\boldsymbol{s},\mathcal{T}) + \mathcal{T}\delta) + (\delta+2\theta)|\mathcal{S}|\mathcal{T}\mathcal{G}_{\mathcal{T}}.
\end{aligned}
$$

Similarly, we could also get inequality in the other direction $\mathcal{V}_i^{\pi^*}(\boldsymbol{s},\mathcal{T}) \geq (1-\frac{\delta}{\theta})^{\mathcal{T}}(\mathcal{V}^{\pi^*}(\boldsymbol{s},\mathcal{T})) - (\delta+2\theta)|\mathcal{S}|\mathcal{T}\mathcal{G}_{\mathcal{T}}$. Here, $V_{\Psi_1}^{\pi^*}(\boldsymbol{s},\mathcal{T})$ and $V_{\Psi_2}^{\pi^*}(\boldsymbol{s},\mathcal{T})$ denote the part of $\mathcal{T}$-step value function which only include paths in Ψ_1 and Ψ_2, respectively.

Let us set θ to be $\sqrt{\delta}$. Note that we could assume $\delta \leq 1$ without loss of generality since we could always rescale the reward function to $[0,1]$. For some $\omega \geq 0$, if $\delta \leq (\frac{\omega}{12|\mathcal{S}|\mathcal{T}\mathcal{G}_{\mathcal{T}}})^2$, $(1+\frac{\delta}{\theta})^{\mathcal{T}}\mathcal{V}^{\pi^*}(\boldsymbol{s},\mathcal{T})$ is no greater than $\frac{\omega}{8}+\mathcal{V}^{\pi^*}(\boldsymbol{s},\mathcal{T})$, and $(1+\frac{\delta}{\theta})^{\mathcal{T}}\mathcal{T}\delta \leq \frac{\omega}{8}$. Combined with $(\delta+2\theta)|\mathcal{S}|\mathcal{T}\mathcal{G}_{\mathcal{T}} \leq \frac{\omega}{4}$, we could get $|\mathcal{V}^{\pi^*}(\boldsymbol{s},\mathcal{T}) - \mathcal{V}_i^{\pi^*}(\boldsymbol{s},\mathcal{T})| \leq \omega$, where $i \in \{1,2\}$, independent of $\boldsymbol{s} \in \mathcal{S}$. □

Lemma 5. *Let the state and state-action value be $V(\boldsymbol{s})$ and $Q(\boldsymbol{s},a)$ respectively, and let the observed state with higher variance of Q value be state $\boldsymbol{s}_{t_1}$ and the observed state with smaller variance of Q value be $\boldsymbol{s}_{t_2}$. The variance is taken over different actions. Let π denote the current policy. Then, we have the following*

$$
\mathbb{E}_\pi[\sum_{t=0}^{T}\gamma^t r_t | do(s_{t_1}=\hat{s}_{t_1})] \leq \mathbb{E}_\pi[\sum_{t=0}^{T}\gamma^t r_t | do(s_{t_2}=\hat{s}_{t_2})],
$$

where $do(\boldsymbol{s}_{t_1}=\tilde{\boldsymbol{s}}_{t_1})$ means the observed state at time step t_1 is perturbed from $\boldsymbol{s}_{t_1}$ to $\tilde{s}_{t_1}$ by using the adversarial perturbation, and $do(s_{t_2}=\tilde{\boldsymbol{s}}_{t_2})$ means that the observed state at time step t_2 is changed from $\boldsymbol{s}_{t_2}$ to $\tilde{\boldsymbol{s}}_{t_2}$ by utilizing the adversarial perturbation.

9.6 Proof of Lemma 4

Proof. For the sake of convenience, we call the non-private version of Algorithm 2 as Pairwise GIGA and denote by $w_t = \mathcal{A}(D)$, $w'_t = \mathcal{A}(D')$. Also, we let $D_t = \{z_1, \cdots, z_t\}$.

We will show that the sensitivity of the t-th iteration in Pairwise GIGA is at most $\frac{8G}{\alpha t}$. We prove it by induction.

We first consider the case $1 \leq t \leq T_1$. Since $w_1, \cdots, w_{T_1}$ are selected randomly, their values do not depend on the underlying dataset. Thus, we have $w_t = w'_t$ for all $1 \leq t \leq T_1$.

Next, we consider $t > T_1$. There are two cases, *i.e.,* $D - D' = \{z_t, z'_t\}$ and $D - D' = \{z_i, z'_i\}$, where $i < t$.

For the first case, since $D - D' = \{z_t, z'_t\}$, we have $w_{t-1} = w'_{t-1}$. Thus

$$\begin{aligned}
&\|w_t - w'_t\|_2 \leq \|w_{t-1} - \eta_t \nabla \hat{L}_t(w_{t-1}, D_t) - w'_{t-1} + \eta_t \nabla \hat{L}_t(w_{t-1}, D'_t)\|_2 \\
&= \eta_t \|\nabla \hat{L}_t(w_{t-1}, D_t) - \nabla \hat{L}_t(w_{t-1}, D'_t)\|_2 \leq \frac{t-1}{t-2} \frac{2G}{\alpha t} \leq \frac{4G}{\alpha t},
\end{aligned}$$

where the last inequality is due to the G-Lipschitz assumption on ℓ and the assumption of $t \geq 3$.

For the second case, we have the following

$$\begin{aligned}
&\|w_t - w'_t\|_2^2 \\
&\leq \|(w_{t-1} - \eta_t \nabla \hat{L}_t(w_{t-1}, D_t)) \\
&\qquad - (w'_{t-1} - \eta_t \nabla \hat{L}_t(w'_{t-1}, D'_t))\|_2^2 \qquad (9.12) \\
&\leq \|w_{t-1} - w'_{t-1}\|_2^2 \\
&\qquad + \eta_t^2 \|\nabla \hat{L}_t(w_{t-1}, D_t) - \nabla \hat{L}_t(w'_{t-1}, D'_t)\|_2^2 \\
&\qquad - 2\eta_t (w_{t-1} - w'_{t-1})^T (\nabla \hat{L}_t(w_{t-1}, D_t) \\
&\qquad - \nabla \hat{L}_t(w'_{t-1}, D'_t)). \qquad (9.13)
\end{aligned}$$

For the term $\|\nabla \hat{L}_t(w_{t-1}, D_t) - \nabla \hat{L}_t(w'_{t-1}, D'_t)\|_2^2$, we have

$$\begin{aligned}
&\|\nabla \hat{L}_t(w_{t-1}, D_t) - \nabla \hat{L}_t(w'_{t-1}, D'_t)\|_2^2 \\
&= \|\frac{1}{t-1}\sum_{j\neq i}[\nabla \ell(w_{t-1}; z_t, z_j) - \nabla \ell(w'_{t-1}; z_t, z_j)] \\
&\quad + \frac{1}{t-1}[\nabla \ell(w_{t-1}; z_t, z_i) - \nabla \ell(w'_{t-1}; z_t, z'_i)]\|_2^2 \\
&\leq 2\|\frac{1}{t-1}\sum_{j\neq i}[\nabla \ell(w_{t-1}; z_t, z_j) - \nabla \ell(w'_{t-1}; z_t, z_j)]\|_2^2 \\
&\quad + 2\|\frac{1}{t-1}[\nabla \ell(w_{t-1}; z_t, z_i) - \nabla \ell(w'_{t-1}; z_t, z'_i)]\|_2^2 \\
&\leq 2L^2(\frac{t-2}{t-1})^2\|w_{t-1} - w'_{t-1}\|_2^2 + \frac{8G^2}{(t-1)^2},
\end{aligned} \tag{9.14}$$

where the last inequality is due to the L-smoothness and G-Lipschitz of the loss function ℓ.

For the term $(w_{t-1} - w'_{t-1})^T(\nabla \hat{L}_t(w_{t-1}, D_t) - \nabla \hat{L}_t(w'_{t-1}, D'_t))$, we have:

$$\begin{aligned}
&(w_{t-1} - w'_{t-1})^T(\nabla \hat{L}_t(w_{t-1}, D_t) - \nabla \hat{L}_t(w'_{t-1}, D'_t)) \\
&= (w_{t-1} - w'_{t-1})^T[\frac{1}{t-1}\sum_{j\neq i}[\nabla \ell(w_{t-1}; z_t, z_j) \\
&- \nabla \ell(w'_{t-1}; z_t, z_j)] \\
&+ \frac{1}{t-1}[\nabla \ell(w_{t-1}; z_t, z_i) - \nabla \ell(w'_{t-1}; z_t, z'_i)].
\end{aligned} \tag{9.15}$$

By the α-strongly convexity of the loss function, we have

$$\begin{aligned}
&(w_{t-1} - w'_{t-1})^T[\frac{1}{t-1}\sum_{j\neq i}[\nabla \ell(w_{t-1}; z_t, z_j) - \nabla \ell(w'_{t-1}; z_t, z_j)] \\
&\geq \alpha\frac{t-2}{t-1}\|w_{t-1} - w'_{t-1}\|_2^2.
\end{aligned} \tag{9.16}$$

Also due to the G-Lipschitz, we have

$$\begin{aligned}
&|(w_{t-1} - w'_{t-1})^T[\frac{1}{t-1}[\nabla \ell(w_{t-1}; z_t, z_i) - \nabla \ell(w'_{t-1}; z_t, z'_i)]| \\
&\leq \frac{2G\|w_{t-1} - w'_{t-1}\|_2}{t-1}.
\end{aligned} \tag{9.17}$$

Plugging (9.16) and (9.17) into (9.15), we have

$$\begin{aligned}
&(w_{t-1} - w'_{t-1})^T(\nabla \hat{L}_t(w_{t-1}, D_t) - \nabla \hat{L}_t(w'_{t-1}, D'_t)) \\
&\quad \geq \alpha\frac{t-2}{t-1}\|w_{t-1} - w'_{t-1}\|_2^2 - \frac{2G\|w_{t-1} - w'_{t-1}\|_2}{t-1}.
\end{aligned} \tag{9.18}$$

Plugging (9.18) and (9.14) into (9.13), we get

$$\|w_t - w'_t\|_2^2 \leq (1 + 2L^2\eta_t^2(\frac{t-2}{t-1})^2 - 2\eta_t\alpha\frac{t-2}{t-1})\|w_{t-1} - w'_{t-1}\|_2^2 + \frac{8G^2\eta_t^2}{(t-1)^2} + \frac{4\eta_t G\|w_{t-1} - w'_{t-1}\|_2}{t-1}. \tag{9.19}$$

Now taking $\eta_t = \frac{t-1}{t-2}\frac{2}{\alpha t}$ and $\|w_{t-1} - w'_{t-1}\|_2 \leq \frac{8G}{\alpha(t-1)}$, we have

$$\begin{aligned}\|w_t - w'_t\|_2^2 &\leq (1 + \frac{8L^2}{\alpha^2 t^2} - \frac{4}{t})\frac{64G^2}{\alpha^2(t-1)^2} + \frac{32G^2}{\alpha^2 t^2(t-2)^2} + \frac{64G^2}{\alpha^2 t(t-1)(t-2)} \\ &\leq (1 + \frac{8L^2}{\alpha^2 t^2} - \frac{4}{t} + \frac{1}{2(t-2)^2} + \frac{1}{(t-2)})\frac{64G^2}{\alpha^2(t-1)^2}. \end{aligned} \tag{9.20}$$

What we still need to prove is

$$(1 + \frac{8L^2}{\alpha^2 t^2} - \frac{4}{t} + \frac{1}{2(t-2)^2} + \frac{1}{(t-2)})\frac{64G^2}{\alpha^2(t-1)^2} \leq \frac{64G^2}{\alpha^2 t^2}. \tag{9.21}$$

After simplifying both sides we now need to show

$$\frac{8L^2}{\alpha^2} + \frac{t^2}{2(t-2)^2} + \frac{t^2}{t-2} \leq 2t + 1. \tag{9.22}$$

By the assumption on $t \geq T_1 = \max\{\frac{16L^2}{\alpha^2}, 7\}$, we have $\frac{t}{2} \geq \frac{8L^2}{\alpha^2}$, $\frac{3}{2}t \geq \frac{t^2}{t-2}$ and $1 \geq \frac{t^2}{2(t-2)^2}$. Thus, (9.22) is true, and we have

$$\|w_t - w'_t\|_2^2 \leq \frac{64G^2}{\alpha^2 t^2}.$$

This completes the proof. □

9.7 Proof of Theorem 6

Proof. By Lemma 4, we know the ℓ_2 norm sensitivity in the t-th iteration is upper bounded by $\frac{8G}{\alpha t}$. Now, by Lemma 4 and Lemma 2 we can get that each iteration of Algorithm 2 is $\frac{\rho}{n-T_1}$-zCDP for $T_1 < t \leq n$. Then by Lemma 1 we can see that Algorithm 2 is ρ-zCDP. Thus by Lemma 3, it is (ϵ, δ)-DP. □

9.8 Proof of Theorem 7

For the sake of convenience, we call the non-private version of Algorithm 2 as Pairwise GIGA and denote by $w_t = \mathcal{A}(D)$, $w'_t = \mathcal{A}(D')$. Also, we let $D_t = \{z_1, \cdots, z_t\}$. As we said earlier, in the case of $\frac{\epsilon}{\log\frac{1}{\delta}} \ll 1$ we can see $\sigma_t^2 = c\frac{\log\frac{1}{\delta}G^2(n-T_1)}{\alpha^2\epsilon^2 t^2} = O(\frac{\log\frac{1}{\delta}G^2 n}{\alpha^2\epsilon^2 t^2})$ for $c = 128$. We first prove the following lemma:

Lemma 6. *Let $\mathcal{R}_{GIGA}(n, D)$ be the regret of (non-private) Pairwise GIGA on the stream $\{z_1, z_2, \cdots, z_n\}$, then the outputs $w_1, \cdots, w_{n-1}$ satisfies*

$$\sum_{t=2}^{n} \hat{L}_t(w_{t-1}, D_t) - \min_{w \in \mathcal{C}} \sum_{t=2}^{n} \hat{L}_t(w, D_t)$$
$$\leq \mathcal{R}_{GIGA}(n, D) + G \sum_{t=T_1+1}^{T} \|n_{t-1}\|_2 + GT_1\|\mathcal{C}\|_2. \tag{9.23}$$

Proof of Lemma 6. We denote the output of Pairwise GIGA as $\tilde{w}_1, \cdots, \tilde{w}_{n-1}$. Then, by the G-Lipschitz property of ℓ and $\hat{L}_t$, we get

$$\sum_{t=2}^{n} \hat{L}_t(w_{t-1}, D_t) - \min_{w \in \mathcal{C}} \sum_{t=2}^{n} \hat{L}_t(w, D_t) \leq \sum_{t=2}^{n} \hat{L}_t(w_{t-1}, D_t) - \sum_{t=2}^{n} \hat{L}_t(\tilde{w}_{t-1}, D_t) + \mathcal{R}_{GIGA}(n, D)$$
$$\leq G \sum_{t=2}^{n} \|w_{t-1} - \tilde{w}_{t-1}\|_2 + \mathcal{R}_{GIGA}(n, D) = \mathcal{R}_{GIGA}(n, D) + G \sum_{t=T_1+1}^{n} \|n_{t-1}\|_2 + GT_1\|\mathcal{C}\|_2.$$

□

Next we bound the term of $\sum_{t=T_1+1}^{T} \|n_{t-1}\|_2$. For a Gaussian distribution $x \sim \mathcal{N}(0, \sigma^2 I_d)$, with probability at least $1 - \zeta$ we have $\|x - \sigma\|_2 \leq \sigma\sqrt{d}\sqrt{2\log 2/\zeta}$. Thus, by the above concentration bound and taking the union, we have the following with probability at least $1 - \zeta$

$$\sum_{t=T_1+1}^{T} \|n_{t-1}\|_2 \leq O(\sum_{t=T_1}^{n} \frac{\sqrt{d}\sqrt{\log \frac{n}{\zeta}} G\sqrt{n - T_1}\sqrt{\log 1/\delta}}{\alpha\epsilon t}) \leq O(\frac{G\sqrt{d}\log^{1.5} \frac{n}{\zeta}\sqrt{n}\sqrt{\log 1/\delta}}{\alpha\epsilon}). \tag{9.24}$$

Combining this with Lemma 6 and (9.24), we can get the following with probability at least $1 - \zeta$

$$\sum_{t=2}^{n} \hat{L}_t(w_{t-1}, D_t) - \min_{w \in \mathcal{C}} \sum_{t=2}^{n} \hat{L}_t(w, D_t) \leq \mathcal{R}_{GIGA}(n) + O(\frac{G^2\sqrt{d}\log^{1.5} \frac{n}{\zeta}\sqrt{n}\sqrt{\log 1/\delta}}{\alpha\epsilon} + \frac{GL^2}{\alpha^2}\|\mathcal{C}\|_2).$$

Using the regret bound analysis of GIGA in [144, 175] on strongly convex functions $\{\hat{L}_t\}_{t=1}^{n}$ and by the fact that they are α-strongly convex, we can get

$$\mathcal{R}_{GIGA}(n, D) \leq \frac{G^2(1 + \log n)}{2\alpha}.$$

Thus, in total we have

$$\sum_{t=2}^{n} \hat{L}_t(w_{t-1}, D_t) - \min_{w \in \mathcal{C}} \sum_{t=2}^{n} \hat{L}_t(w, D_t) \leq O(\frac{G^2\sqrt{d}\log^{1.5} \frac{n}{\zeta}\sqrt{n}\sqrt{\log 1/\delta}}{\alpha\epsilon}$$
$$+ \frac{GL^2}{\alpha^2}\|\mathcal{C}\|_2 + \frac{G^2(1 + \log n)}{\alpha}).$$

□

9.9 Proof of Theorem 8

By the perturbation strategy in Algorithm 3 and Theorem 7 we can get the following for the loss function after perturbation $\tilde{\ell} = \ell + \frac{\alpha}{2}\|w - w_0\|^2$

$$\mathcal{R}_{\mathcal{A},\tilde{\ell}}(n, D) \leq O(\frac{(G + \alpha\|\mathcal{C}\|_2)^2\sqrt{d}\log^{1.5}\frac{n}{\zeta}\sqrt{n}\sqrt{\log 1/\delta}}{\alpha\epsilon} + \frac{(G + \alpha\|\mathcal{C}\|_2)(L + \alpha)^2}{\alpha^2}\|\mathcal{C}\|_2 + \frac{(G + \alpha\|\mathcal{C}\|_2)^2(1 + \log n)}{\alpha}).$$

Since $\mathcal{R}_{\mathcal{A},\ell}(n, D) \leq \mathcal{R}_{\mathcal{A},\tilde{\ell}}(n, D) + n\alpha\|\mathcal{C}\|_2^2$, we have

$$\mathcal{R}_{\mathcal{A},\ell}(n, D) \leq O(\frac{(G + \alpha\|\mathcal{C}\|_2)^2\sqrt{d}\log^{1.5}\frac{n}{\zeta}\sqrt{n}\sqrt{\log 1/\delta}}{\alpha\epsilon} + \frac{(G + \alpha\|\mathcal{C}\|_2)(L + \alpha)^2}{\alpha^2}\|\mathcal{C}\|_2 + \frac{(G + \alpha\|\mathcal{C}\|_2)^2(1 + \log n)}{\alpha} + n\alpha\|\mathcal{C}\|_2^2).$$

Taking $\alpha = O(\frac{1}{\sqrt[4]{n}})$, we get

$$\mathcal{R}_{\mathcal{A},\ell}(n, D) \leq O(\frac{L^2G^2\|\mathcal{C}\|_2^2\sqrt{d}\log^{1.5}\frac{n}{\zeta}n^{\frac{3}{4}}\sqrt{\log 1/\delta}}{\epsilon}).$$

□

9.10 Proof of Theorem 9

We first rephrase a lemma in [129]. We denote the Rademacher averages to the pairwise loss functions class $\ell \circ \mathcal{C} := \{(z, z') \mapsto \ell(w; z, z'), w \in \mathcal{C}\}$ as the following:

$$\mathcal{R}_n(\ell \circ \mathcal{C}) = \mathbb{E}[\sup_{w\in\mathcal{C}} \frac{1}{n}\sum_{i=1}^{n} \epsilon_i \ell(w; , z, z_i)]$$

where $\{\epsilon_i\}_{i=1}^n$ are Rademacher variables, *i.e.,* $\epsilon = \pm 1$ with probability $\frac{1}{2}$. And the expectation is over $\{\epsilon_i\}_{i=1}^n, z, \{z_i\}_{i=1}^n$.

Lemma 7 (Theorem 3 in [129])**.** *Let $w_1, \cdots, w_{n-1}$ be an ensemble of parameters generated by an online learning algorithm working with a B-bounded pairwise loss function ℓ that guarantees a regret bound of $\mathcal{R}(n)$. Then for any $\delta > 0$, we have the following with probability at least $1 - \delta$,*

$$L_{\mathcal{P}}(\bar{w}) \leq \frac{1}{n-1}\sum_{t=2}^{n} L_{\mathcal{P}}(w_{t-1}) \leq \min_{w\in\mathcal{C}} L_{\mathcal{P}}(w) + \frac{4}{n-1}\sum_{t=2}^{n}\mathcal{R}_{t-1}(\ell \circ \mathcal{C}) + \frac{\mathcal{R}(n)}{n-1} + 6B\sqrt{\frac{\log\frac{n}{\delta}}{n-1}}. \tag{9.25}$$

For the strongly convex loss functions, Theorem 7 and Lemma 7, we can get with probability at least $1-2\zeta$,

$$L_{\mathcal{P}}(\bar{w}) - \min_{w\in\mathcal{C}} L_{\mathcal{P}}(w) \leq O\big(\frac{1}{n-1}\sum_{t=2}^{n}\mathcal{R}_{t-1}(\ell\circ\mathcal{C}) + \frac{G^2L^2\|\mathcal{C}\|_2\sqrt{d}\log^{1.5}\frac{n}{\zeta}\sqrt{\log 1/\delta}}{\alpha^2\epsilon\sqrt{n}}\big). \tag{9.26}$$

For the general convex ones, we have with probability at least $1-2\zeta$

$$\begin{aligned} L_{\mathcal{P}}(\bar{w}) - \min_{w\in\mathcal{C}} L_{\mathcal{P}}(w) &\leq O\big(\frac{1}{n-1}\sum_{t=2}^{n}\mathcal{R}_{t-1}(\ell\circ\mathcal{C}) + G\|\mathcal{C}\|_2\sqrt{\frac{\log\frac{n}{\zeta}}{n-1}} \\ &\quad + \frac{L^2G^2\|\mathcal{C}\|_2^2\sqrt{d}\log^{1.5}\frac{n}{\zeta}n^{\frac{3}{4}}\sqrt{\log 1/\delta}}{\epsilon(n-1)}\big) \\ &= O\big(\frac{1}{n-1}\sum_{t=2}^{n}\mathcal{R}_{t-1}(\ell\circ\mathcal{C}) + \frac{L^2G^2\|\mathcal{C}\|_2^2\sqrt{d}\log^{1.5}\frac{n}{\zeta}\sqrt{\log 1/\delta}}{\epsilon\sqrt[4]{n}}\big). \end{aligned}$$

□

9.11 Proof of Theorem 10

We first prove Algorithm 4 is (ϵ,δ) differentially private. What we only need to show is the sensitivity of $\tilde{w}$ is $\frac{8G\log n}{n\alpha}$. Since by Lemma 4, we know $\|w_t - w_t'\|_2 \leq \frac{8G}{\alpha t}$, thus

$$\|\bar{w} - \bar{w}'\|_2 \leq \frac{\sum_{t=1}^{n}\frac{8G}{\alpha t}}{n} \leq \frac{8G\log n}{n\alpha}. \tag{9.27}$$

Thus by Gaussian mechanism we can show it is (ϵ,δ)-differentially private.

Next we analyze the generalization error, we have the following with probability $1-\zeta$:

$$L_{\mathcal{P}}(\hat{w}) - \min_{w\in\mathcal{C}} L_{\mathcal{P}}(w) \leq L_{\mathcal{P}}(\hat{w}) - L_{\mathcal{P}}(\tilde{w}) + L_{\mathcal{P}}(\tilde{w}) - \min_{w\in\mathcal{C}} L_{\mathcal{P}}(w) \tag{9.28}$$

$$\leq G\|\hat{w}-\tilde{w}\|_2 + \frac{4}{n-1}\sum_{t=2}^{n}\mathcal{R}_{t-1}(\ell\circ\mathcal{C}) + \frac{\mathcal{R}_{GIGA}(n,D)}{n} + 6G\|\mathcal{C}\|_2\sqrt{\frac{\log\frac{n}{\zeta}}{n}},$$

$$\leq G\|\hat{w}-\bar{w}\|_2 + G\|\bar{w}-\tilde{w}\|_2 + \frac{4}{n-1}\sum_{t=2}^{n}\mathcal{R}_{t-1}(\ell\circ\mathcal{C})$$

$$+ \frac{\mathcal{R}_{GIGA}(n,D)}{n} + 6G\|\mathcal{C}\|_2\sqrt{\frac{\log\frac{n}{\zeta}}{n}}, \tag{9.29}$$

where $\mathcal{R}_{GIGA}(n, D)$ is the regret of Pairwise GIGA on the strongly convex loss function $\{\hat{L}_t\}_{t=1}^n$. The last inequality is by the G-Lipschitz property and Lemma 7 of this supplementary material.

Also, by [175], the regret of Pairwise GIGA on the strongly convex loss function $\{\hat{L}_t\}_{t=1}^n$ is $\mathcal{R}_{GIGA}(n, D) \leq \frac{2G^2(1+\log n)}{\alpha}$. For the term $\|\hat{w} - \bar{w}\|_2$, by definition of $\hat{w}$, we have

$$\|\hat{w} - \bar{w}\|_2 \leq \|\tilde{w} - \bar{w}\|_2.$$

For the term $\|\bar{w} - \tilde{w}\| = \|\sigma\|$, we have with probability at least $1 - \zeta$,

$$\|\sigma\|_2 \leq \frac{8G\sqrt{d}\sqrt{2}\sqrt{\log 1/\zeta \log 1.25/\delta} \log n}{\alpha n \epsilon}.$$

Thus in total we have:

$$L_{\mathcal{P}}(\bar{w}) - \min_{w \in \mathcal{C}} L_{\mathcal{P}}(w) \leq O\big(\frac{\sqrt{d}G^2\sqrt{\log 1/\zeta \log 1/\delta} \log n}{\alpha n \epsilon} + \frac{1}{n}\sum_{t=1}^{n} \mathcal{R}_t(\ell \circ \mathcal{C}) + \frac{G^2 \log n}{\alpha n} + G\|\mathcal{C}\|_2\sqrt{\frac{\log \frac{n}{\zeta}}{n}}\big).$$

For the convex loss function, as the same as above, we have

$$L_{\mathcal{P}}(\bar{w}) - \min_{w \in \mathcal{C}} L_{\mathcal{P}}(w) \leq O\big(\frac{\sqrt{d}(G + \alpha\|\mathcal{C}\|_2)^2\sqrt{\log 1/\zeta \log 1/\delta} \log n}{\alpha n \epsilon} + \frac{1}{n}\sum_{t=1}^{n} \mathcal{R}_t(\ell \circ \mathcal{C}) + \frac{(G + \alpha\|\mathcal{C}\|_2)^2 \log n}{\alpha n} + (G + \alpha\|\mathcal{C}\|_2)\|\mathcal{C}\|_2\sqrt{\frac{\log \frac{n}{\zeta}}{n}} + \alpha\|\mathcal{C}\|_2^2\big).$$

When we take $\alpha = O(\frac{1}{\sqrt{n}})$, we have

$$L_{\mathcal{P}}(\bar{w}) - \min_{w \in \mathcal{C}} L_{\mathcal{P}}(w) \leq O\big(\frac{\sqrt{d}G^2\|\mathcal{C}\|_2^2 \log \frac{n}{\zeta}\sqrt{\log 1/\delta} \log n}{\sqrt{n}\epsilon} + \frac{1}{n}\sum_{t=1}^{n} \mathcal{R}_t(\ell \circ \mathcal{C})\big). \quad (9.30)$$

□

9.12 Proof of Theorem 11

In this section, we aim to prove Theorem 1, of which the proof procedure will utilize the results of Lemma 1 and 2. Let $T_1 = \int(\widetilde{f}_i(x) - f_i(x))f_i(x)dx$, and $T_2 = \int(E[\widetilde{f}_i^2(x)] - f_i^2(x))dx$. Then, we have

$\Delta_{f_i} = E[\int(f_i(x) - \widetilde{f}_i(x))^2 dx]$

$= \int f_i^2(x)dx - 2\int E[\widetilde{f}_i(x)]f_i(x)dx + \int \widetilde{f}_i^2(x)dx$
$= \int f_i^2(x)dx - 2\int E([\widetilde{f}_i(x)] - f_i(x))f_i(x)dx$
$-2\int f_i^2(x)dx + \int(\widetilde{f}_i^2(x) - f_i^2(x))dx + \int f_i^2(x)dx$
$= -2\int E([\widetilde{f}_i(x)] - f_i(x))f_i(x)dx + \int(E[\widetilde{f}_i^2(x)] - f_i^2(x))$
$= -2T_1 + T_2,$

Since $\Delta_{f_i} = E[\int(f_i(x) - \widetilde{f}_i(x))^2dx] \geq 0$, we have $\Delta_{f_i} = -2T_1 + T_2 = |-2T_1 + T_2| \leq |-2T_1| + |T_2| = |2T_1| + |T_2|$.

Combining the results of Lemmas 1 and 2 which provide the upper bounds for T_1 and T_2, gives us the followings result.

$$\Delta_{f_i} \leq |2T_1| + |T_2| \leq 2A_i\widetilde{h}_i + 2A_i\widetilde{h}_i + A_i^2\widetilde{h}_i^2V_i + \frac{\rho B}{|\widetilde{\mathcal{S}}_i|\widetilde{h}_i^d} + \frac{\rho A_i BV_i}{|\widetilde{\mathcal{S}}_i|\widetilde{h}_i^{d-1}}$$
$$= 4A_i\widetilde{h}_i + A_i^2\widetilde{h}_i^2V_i + \frac{\rho B}{|\widetilde{\mathcal{S}}_i|\widetilde{h}_i^d} + \frac{\rho A_i BV_i}{|\widetilde{\mathcal{S}}_i|\widetilde{h}_i^{d-1}}.$$

Thus, it is clear that f_i and $\widetilde{f}_i$ will be close if $|\widetilde{\mathcal{S}}_i|$ is large enough and if $\widetilde{h}_i$ is chosen properly as a function of $|\widetilde{\mathcal{S}}_i|$.

Below we will denote $\widetilde{x}_{i,s} = \frac{1}{|\widetilde{\mathcal{S}}_i|}$.

Lemma 8. *Let T_1 be $T_1 = \int E([\widetilde{f}_i(x)] - f_i(x))f_i(x)dx$. Then, $|T_1| \leq A_i\widetilde{h}_i$.*

Proof. $T_1 = \int E([\widetilde{f}_i(x)] - f_i(x))f_i(x)dx$
$= \int(E[\sum_{s\in\widetilde{\mathcal{S}}_i}\widetilde{l}_{i,s}\mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - \widetilde{x}_{i,s})] - f_i(x))f_i(x)dx$
$= \int(\sum_{s\in\widetilde{\mathcal{S}}_i}\widetilde{l}_{i,s}\int\mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - \widetilde{x}_{i,s})f_i(\widetilde{x}_{i,s})d\widetilde{x}_{i,s} - f_i(x))f_i(x)dx$
$\leq \int(\sum_{s\in\widetilde{\mathcal{S}}_i}\widetilde{l}_{i,s}|\int\mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - \widetilde{x}_{i,s})f_i(\widetilde{x}_{i,s})d\widetilde{x}_{i,s} - f_i(x)|)f_i(x)dx$
$\leq \int(\sum_{s\in\widetilde{\mathcal{S}}_i}\widetilde{l}_{i,s}A_i\widetilde{h}_i)f_i(x)dx = \int(A_i\widetilde{h}_i)f_i(x)dx = A_i\widetilde{h}_i.$
Lemma B.3 and that $\int f_i(x)dx = 1$ are used in the last line. □

Lemma 9. *Let T_2 be as in above, and let V_i be the volume of the support of f_i. Then, $T_2 \leq 2A_i\widetilde{h}_i + A_i^2\widetilde{h}_i^2V_i + \frac{\rho B}{|\widetilde{\mathcal{S}}_i|\widetilde{h}_i^d} + \frac{\rho A_i BV_i}{|\widetilde{\mathcal{S}}_i|\widetilde{h}_i^{d-1}}$.*

Proof. Since $\widetilde{f}_i(x) = \sum_{s\in\widetilde{\mathcal{S}}_i}\widetilde{l}_{i,s}\mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - \widetilde{x}_{i,s})$, then $\widetilde{f}_i^2(x) = \sum_{s\neq t}\widetilde{l}_{i,s}\widetilde{l}_{i,t}\mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - \widetilde{x}_{i,s})\mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - \widetilde{x}_{i,t}) + \sum_s\widetilde{l}_{i,s}^2\mathcal{K}_{\widetilde{\mathcal{H}}_i}^2(x - \widetilde{x}_{i,s})$. For convenience, we drop the $\widetilde{\mathcal{H}}_i$ subscript and denote $\mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - \widetilde{x}_{i,s})$ as $\mathcal{K}(\widetilde{x}_{i,s})$, Then, T_2 can be simplified as follows:

$T_2 = \int(E[\widetilde{f}_i^2(x)] - f_i^2(x))dx$
$= \int(E[\sum_{s\neq t}\widetilde{l}_{i,s}\widetilde{l}_{i,t}\mathcal{K}(\widetilde{x}_{i,s})\mathcal{K}(\widetilde{x}_{i,t}) + \sum_s\widetilde{l}_{i,s}^2\mathcal{K}^2(\widetilde{x}_{i,s})] - f_i^2(x))dx$
$= \int(\sum_{s\neq t}\widetilde{l}_{i,s}\widetilde{l}_{i,t}\int\int\mathcal{K}(\widetilde{x}_{i,s})\mathcal{K}(\widetilde{x}_{i,t})f(\widetilde{x}_{i,s})f(\widetilde{x}_{i,t})d\widetilde{x}_{i,s}d\widetilde{x}_{i,t}$
$+ \sum_s\widetilde{l}_{i,s}^2\int\mathcal{K}^2(\widetilde{x}_{i,s})f_i(\widetilde{x}_{i,s})d\widetilde{x}_{i,s} - \sum_{s\neq t}\widetilde{l}_{i,s}\widetilde{l}_{i,t}f_i^2(x) - \sum_s\widetilde{l}_{i,s}^2f_i^2(x))dx$
$= \int(\sum_{s\neq t}\widetilde{l}_{i,s}\widetilde{l}_{i,t}[\int\int\mathcal{K}(\widetilde{x}_{i,s})\mathcal{K}(\widetilde{x}_{i,t})f(\widetilde{x}_{i,s})f(\widetilde{x}_{i,t})d\widetilde{x}_{i,s}d\widetilde{x}_{i,t}$

$-f_i^2(x)] + \sum_s \widetilde{l}_{i,s}^2 \int \mathcal{K}^2(\widetilde{x}_{i,s}) f_i(\widetilde{x}_{i,s}) d\widetilde{x}_{i,s} - \sum_s \widetilde{l}_{i,s}^2 f_i^2(x)) dx$
$\leq \int (\sum_{s \neq t} \widetilde{l}_{i,s} \widetilde{l}_{i,t} (2A_i \widetilde{h}_i f_i(x) + A_i^2 \widetilde{h}_i^2) + \sum_s \widetilde{l}_{i,s}^2 B(f_i(x) + A_i \widetilde{h}_i) \widetilde{h}_i^{-d}) dx$
$= 2A_i \widetilde{h}_i + A_i^2 \widetilde{h}_i^2 V_i + \frac{\rho B}{|\widetilde{\mathcal{S}}_i| \widetilde{h}_i^d} + \frac{\rho A_i B V_i}{|\widetilde{\mathcal{S}}_i| \widetilde{h}_i^{d-1}}$.

In the above, we have made use of Lemmas B.4 and the fact that $\sum_{s \neq t} \widetilde{l}_{i,s} \widetilde{l}_{i,t} \leq 1$. And, $0 \leq \rho \leq |\widetilde{\mathcal{S}}_i|$. □

Lemma 10. *For $\mathcal{O}_i$, let $A_i = sup_{x \in R} \|\nabla f_i(x)\|$. Then $\|\int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x-y) f_i(y) dy - f_i(x)\| \leq A_i \widetilde{h}_i$.*

Proof. The proof of this lemma contains the following two steps. Firstly, let $x, y \in \mathbb{R}$ such that $\|x - y\| \leq \widetilde{h}_i$. And, $A_i = sup_{x \in \mathbb{R}} \|\nabla f_i(x)\|$. Define a function $\beta : [0, 1] \to \mathbb{R}$, $\beta(t) = (1-t)x + ty$. Then, we have

$|f_i(y) - f_i(x)| = |f_i(s(1)) - f_i(\beta(0))| = |\int_0^1 \frac{d}{dt} f_i(\beta(t)) dt|$
$= |\int_0^1 \nabla f_i(\beta(t)) \cdot \beta'(t) dt| = |\int_0^1 \nabla f_i(\beta(t)) \cdot (y - x) dt|$
$\leq |\int_0^1 \nabla f_i(\beta(t)) \cdot (y - x) dt| \leq |\int_0^1 \nabla f_i(\beta(t)) \cdot (y - x) dt|$
$\leq \int_0^1 A_i \widetilde{h}_i dt = A_i \widetilde{h}_i$,

In the above, we have made use of the Holder and Cauchy-Schwarz inequalities. Based on this, we then have

$|\int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - y) f_i(y) dy - f_i(x)| = |\int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - y)[f_i(y) - f_i(x)] dy|$
$\leq \int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - y) |f_i(y) - f_i(x)| dy \leq \int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - y) A_i \widetilde{h}_i dy = A_i \widetilde{h}_i$.

In the above, the fact that $\int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(\cdot) = 1$ is used. We have also made use of the fact that in the support of $\mathcal{K}_{\widetilde{\mathcal{H}}_i}(\cdot)$, we have that $\|y - x\| \leq \widetilde{h}_i$ and therefore that $|f_i(y) - f_i(x)| \leq A_i \widetilde{h}_i$ applies. □

Lemma 11. *We have*

$$|\int \int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - y) K_{\widetilde{\mathcal{H}}_i}(x - z) f(y) f(z) dy dz - f_i^2(x)| \leq 2A \widetilde{h}_i f_i(x) + A_i \widetilde{h}_i^2, \quad (9.31)$$

$$\int \mathcal{K}_{\widetilde{\mathcal{H}}_i}^2(x - y) f_i(y) dy \leq B(f_i(x) + A_i \widetilde{h}_i) \widetilde{h}_i^{-d}. \quad (9.32)$$

Proof. Firstly, the proof procedure of (9.31) is as follows.

$= |(\int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - y) f_i(y) dy)(\int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - z) f_i(z) dz) - f_i^2(x)|$
$= |(\int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - y) f_i(y) dy)^2 - f_i^2(x)|$
$= |(\int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - y) f_i(y) dy - f_i(x))(\int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x - y) f_i(y) dy + f_i(x))|$.
$\leq A_i \widetilde{h}_i (2 f_i(x) + A_i \widetilde{h}_i) = 2A_i \widetilde{h}_i f_i(x) + A_i^2 \widetilde{h}_i^2$.

In the above product, the upper bound of the first term is $A_i\widetilde{h}_i$. The second term is upper bounded by $2f_i(x)+A_i\widetilde{h}_i$ since we can get $\int \mathcal{K}_{\widetilde{\mathcal{H}}_i}(x-y)f_i(y)dy \leq f_i(x)+A_i\widetilde{h}_i$ from Lemma 3.

Secondly, we will prove (9.32). Based on Lemma 3, we know that within the integrand's support, $|f_i(y)-f_i(x)| \leq A_i\widetilde{h}_i$, so that $f_i(y) \leq f_i(x)+A_i\widetilde{h}_i$. Also, $\mathcal{K}_{\widetilde{\mathcal{H}}_i}(z) = \widetilde{h}_i^{-1}\mathcal{K}(\widetilde{h}_i^{-1}z)$. Then, we have

$\int \mathcal{K}^2_{\widetilde{\mathcal{H}}_i}(x-y)f_i(y)dy \leq \int \mathcal{K}^2_{\widetilde{\mathcal{H}}_i}(x-y)(f_i(x)+A_i\widetilde{h}_i)dy$
$= (f_i(x)+A_i\widetilde{h}_i)\int \mathcal{K}^2(z)dz = (f_i(x)+A_i\widetilde{h}_i)\widetilde{h}_i^{-1}\int \widetilde{h}_i^{-1}\mathcal{K}^2(\widetilde{h}_i^{-1}z)dz$
$= (f_i(x)+A_i\widetilde{h}_i)\widetilde{h}_i^{-1}\int \mathcal{K}^2(w)dw = B(f_i(x)+A_i\widetilde{h}_i)\widetilde{h}_i^{-d}$.
In the above, a change of variables is used, i.e., $w = \widetilde{h}_i^{-1}I_d z$. □

9.13 Proof of Theorem 12

We consider two neighbor datasets D and D' with size n, we assume $D = \{x_1, x_2, \cdots, x_n\}$ and $D' = \{x_1, x_2, \cdots, x_{n-1}, x'_n\}$, that is $D' = D - \{x_n\} \bigcup \{x'_n\}$. We denote $P_d(y) = Pr\{y = \mathcal{M}(d)\}$ for data record d and the data universe as $\mathcal{Z}$. Also we denote the Algorithm 1 as $\mathcal{F}$. Our will show the differential privacy for $\mathcal{F}$ following [160].

Now we fix y, where $y \in \mathcal{Z}$. The records in a dataset D^* can be partitioned by $I_d(y)$, where $I_d(y)$ is the unique integer which satisfies $\gamma^{-I_d(y)-1} < P_d(y) \leq \gamma^{-I_d(y)}$. If $P_d(y) = 0$, then we denote $I_d(y)$ as $\emptyset$, thus we can define the partition set for i as $C_i(D^*, y) = \{d : d \in D^*, I_d(y) = i\}$.

Lemma 12. *For any dataset D^*, if the seed is in partition set for i. The the probability of passing the privacy test is giving by: $pt(D^*, i, y) = Pr\{L \geq k - |C_i(D^*, y)|\}$ where $L \sim Lap(\frac{1}{\epsilon_0})$.*

Now we assume x_n falls in partition set of j while x'_n falls in partition set of k(if either of them falls into $\emptyset$ the proof is the same), W.L.O.G we assume $j \neq k$(the same proof for $j = k$). We now denote $p(s)$ is the probability we choose the i-th data record, $s \in D$ as the seed of generating the synthetic reord. Thus, it is only depend on the position of the data in D and not dependent on the data record it self, in Algorithm 1, it is just $p(x_{i,j}) = c_{i,j}$

Lemma 13. *For any dataset D^* and data record $y \in \mathcal{Z}$, we have*

$$Pr\{\mathcal{F} = y\} = \sum_{i \geq 0}\Big(\sum_{s \in C_i(D^*, y)} p(s)P_s(y)\Big)pt(D^*, i, y). \tag{9.33}$$

Proof.

$$Pr\{\mathcal{F} = y\} = \sum_{s \in D^*} Pr\{\text{s is the seed,}\mathcal{F}(D^*) = y\} \tag{9.34}$$

$$= \sum_{s} Pr\{\text{select s as the seed}\} Pr\{(D^*, s, y) \text{ passes the test}\} \tag{9.35}$$

$$= \sum_{s} p(s) P_s(y) Pr\{\text{y passes the test}\} \tag{9.36}$$

$$= \sum_{i \geq 0} (\sum_{s \in C_i(D^*, y)} p(s) P_s(y)) pt(D^*, i, y) \tag{9.37}$$

□

We now denote

$$q(D^*, i, y) = (\sum_{s \in C_i(D^*, y)} p(s) P_s(y)) pt(D^*, i, y).$$

Lemma 14. *For D', D as above,*

$$e^{-\epsilon_0} pt(D, i, y) \leq pt(D', i, y) \leq e^{\epsilon_0} pt(D, i, y). \tag{9.38}$$

Proof. If $i \neq j, k$, that means $C_i(D, y) = C_i(D', y)$, so we have $pt(D, i, y) = pt(D', i, y)$.

If $i = j$, that means $|C_i(D', y)| = |C_i(D, y)| + 1$(since x'_n falls in), by the property of Laplace distribution we have:

$$\begin{aligned} pt(D, i, y) &= Pr\{L \geq k - |C_i(D, j)|\} \\ &\leq Pr\{L \geq k - |C_i(D, y)| - 1\} \\ &\leq e^{\epsilon_0} Pr\{L \geq k - |C_i(D, j)|\} \end{aligned}$$

That is $pt(D, i, y) \leq pt(D', i, y) \leq e^{\epsilon_0} pt(D, i, y)$.

If $i = k$, we have the same $e^{-\epsilon_0} pt(D, i, y) \leq pt(D', i, y) \leq pt(D, i, y)$. Thus in total we have

$$e^{-\epsilon_0} pt(D, i, y) \leq pt(D', i, y) \leq e^{\epsilon_0} pt(D, i, y).$$

□

Lemma 15. *For all $i \neq j, k$*

$$q(D, i, y) = q(D', i, y) \tag{9.39}$$

For $i = j$, we have

$$q(D, i, y) \leq q(D', i, y) \tag{9.40}$$

Furthermore, if $|C_j(D, y)| \leq t$,

$$q(D', i, y) \leq te^{-\epsilon_0(k-t)} \max p(s) \tag{9.41}$$

Otherwise,

$$q(D', i, y) \leq e^{\epsilon_0}[1 + \frac{r \max p(s)}{t \min p(s)}]q(D, i, y) \tag{9.42}$$

For $i = k$, we have when $|C_k(D, y)| > t$

$$q(D, i, y) \leq e^{\epsilon_0}[1 + \frac{r \max p(s)}{t \min p(s)}]q(D', i, y) \tag{9.43}$$

Otherwise $q(D, k, y) \leq te^{-\epsilon_0(k-t)} \max p(s)$ and also

$$q(D', i, y) \leq q(D, i, y) \tag{9.44}$$

Proof. (9.39) is oblivious since for $i \neq j, k$ we have $C_i(D, y) = C_i(D', y)$, also by the proof of Lemma 14 we have $ptt(D, i, y) = pt(D', i, y)$. When $i = j$, we have

$$\begin{aligned}
&q(D, i, y) \\
&= pt(D, i, y) \sum_{s \in C_i(D,y)} p(s)P_s(y) \\
&\leq pt(D', i, y)(\sum_{s \in C_i(D,y)} p(s)P_s(y) + p(x'_n)P_{x'_n}(y)) \\
&= q(D', i, y)
\end{aligned}$$

If $|C_i(D, y)| < t$, since $pt(D', i, y) = Pr\{L \geq k - |C_i(D', y)|\} \leq Pr\{L \geq k - t\} = \frac{1}{2}e^{-\epsilon_0(k-t)}$, we have $q(D', i, y) \leq e^{-\epsilon_0(k-t)} \sum_{s \in C_i(D', y)} P_s(y)p(s) \leq te^{-\epsilon_0(k-t)} \max p(s)$, which is (9.41). If $|C_i(D, y)| \geq t$, then we have

$$\begin{aligned}
&q(D', i, y) \\
&= pt(D', i, y) \sum_{s \in C_i(D',y)} p(s)P_s(y) \\
&\leq e^{\epsilon_0} pt(D, i, y)(\sum_{s \in C_i(D,y)} p(s)P_s(y) + p(x'_n)P_{x'_n}(y)) \qquad (9.45)
\end{aligned}$$

By definition, we know $P_{x'_n}(y) \leq \gamma P_s(y)$ for every $s \in C_i(D, y)$, so we have

$$\begin{aligned} P_{x'_n}(y) &\leq \frac{r}{|C_i(D,y)|} \sum_{s \in C_i(D,y)} \gamma P_s(y) \\ &\leq \frac{r}{t} \sum_{s \in C_i(D,y)} \gamma P_s(y) \end{aligned}$$

Also, by the definition of $p(s)$, we have $\frac{p(x'_n)}{p(s)} = \frac{p(x_n)}{p(s)} \leq \frac{\max p(s)}{\min p(s)}$. Thus, we have the following

$$p(x'_n) P_{x'_n} \leq \frac{\gamma \max p(s)}{t \min p(s)} \sum_{s \in C_i(D,y)} p(s) P_s(y) \tag{9.46}$$

Take it into (9.45), we have when $q(D', i, y) \leq e^{\epsilon_0}[1 + \frac{\gamma \max p(s)}{t \min p(s)}] pt(D, i, y)(\sum_{s \in C_i(D,y)} p(s) P_s(y)) = e^{\epsilon_0}[1 + \frac{\gamma \max p(s)}{t \min p(s)}] q(D, i, y)$, which is (9.42).

When $i = k$, it is the same as $i = j$ as we just change the role of D, D', by assumption we have $C_k(D', y) = C_k(D, y) - \{x_n\}$, so we have when $|C_k(D, y)| > t$,

$$\begin{aligned} & q(D, k, y) \\ &= pt(D, k, y) \sum_{s \in C_k(D,y)} p(s) P_s(y) \\ &= pt(D, k, y)(\sum_{s \in C_k(D',y)} p(s) P_s(y) + p(x_n) P_{x_n}(y)) \\ &\leq e^{\epsilon_0} pt(D', k, y)[1 + \frac{\gamma \max p(s)}{t \min p(s)}] \sum_{s \in C_k(D',y)} p(s) P_s(y) \\ &= e^{\epsilon_0} pt(D', k, y)[1 + \frac{\gamma \max p(s)}{t \min p(s)}] q(D', k, y) \end{aligned} \tag{9.47}$$

When $|C_k(D, y)| \leq t$,

$$\begin{aligned} & q(D, k, y) \\ &= pt(D, k, y) \sum_{s \in C_k(D,y)} p(s) P_s(y) \\ &\leq t e^{-\epsilon_0 (k-t)} \max p(s) \end{aligned} \tag{9.48}$$

Others are the same as $i = j$, we omit the proof. □

The following is the key lemma similar with Lemma 5 in [160].

Lemma 16. *For parameters $k \geq 1$, $\gamma > 1$ and $\epsilon_0 > 0$. Take dataset D, D' as above. Then for ant integer $1 \leq t < k$ and synthetic record $y \in \mathcal{Z}$, we have: $Pr\{\mathcal{F}(D)\} \leq e^{\epsilon_0}[1 +$*

$\frac{r \max p(s)}{t \min p(s)}]Pr\{\mathcal{F}(D')\}+te^{-\epsilon_0(k-t)} \max p(s)$, *and* $Pr\{\mathcal{F}(D')\} \leq e^{\epsilon_0}[1+\frac{r \max p(s)}{t \min p(s)}]Pr\{\mathcal{F}(D)\}+ te^{-\epsilon_0(k-t)} \max p(s)$.

Proof. By definition we have: $Pr\{\mathcal{F}(D)\} = \sum_{i \neq j,k} q(D,i,y) + q(D,j,y) + q(D,k,y)$, and $Pr\{\mathcal{F}(D')\} = \sum_{i \neq j,k} q(D',i,y) + q(D',j,y) + q(D',k,y)$. Since $\sum_{i \neq j,k} q(D,i,y) = \sum_{i \neq j,k} q(D',i,y)$ and also by Lemma 15, we have $q(D,j,y) \leq q(D',i,y)$ and $q(D,k,y) \leq e^{\epsilon_0}[1 + \frac{r \max p(s)}{t \min p(s)}]q(D',i,y) + te^{-\epsilon_0(k-t)} \max p(s)$. Thus, we have the following

$$Pr\{\mathcal{F}(D)\} \leq e^{\epsilon_0}[1 + \frac{r \max p(s)}{t \min p(s)}]Pr\{\mathcal{F}(D')\} + te^{-\epsilon_0(k-t)} \max p(s).$$

□

The later proof of Theorem 2 is the same as in [160], we omit them.

References

[1] Martin Boissier, Siwei Lyu, Yiming Ying, and Ding-Xuan Zhou. Fast convergence of online pairwise learning algorithms. In *Artificial Intelligence and Statistics*, pages 204–212, 2016.

[2] Liuyi Yao, Sheng Li, Yaliang Li, Mengdi Huai, Jing Gao, and Aidong Zhang. Representation learning for treatment effect estimation from observational data. In *Advances in Neural Information Processing Systems*, pages 2633–2643, 2018.

[3] Mengdi Huai, Chenglin Miao, Yaliang Li, Qiuling Suo, Lu Su, and Aidong Zhang. Metric learning from probabilistic labels. In *Proc. of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1541–1550, 2018.

[4] Yiming Ying, Longyin Wen, and Siwei Lyu. Stochastic online auc maximization. In *Advances in neural information processing systems*, pages 451–459, 2016.

[5] Michael Natole, Yiming Ying, and Siwei Lyu. Stochastic proximal algorithms for auc maximization. In *Proceedings of the International Conference on Machine Learning*, 2018.

[6] Mengdi Huai, Hongfei Xue, Chenglin Miao, Liuyi Yao, Lu Su, Changyou Chen, and Aidong Zhang. Deep metric learning: the generalization analysis and an adaptive algorithm. In *Proc. of the 28th International Joint Conference on Artificial Intelligence*, 2019.

[7] Qiuling Suo, Weida Zhong, Fenglong Ma, Yuan Ye, Mengdi Huai, and Aidong Zhang. Multi-task sparse metric learning for monitoring patient similarity progression. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 477–486, 2018.

[8] Mengdi Huai, Chenglin Miao, Qiuling Suo, Yaliang Li, Jing Gao, and Aidong Zhang. Uncorrelated patient similarity learning. In *Proc. of the 2018 SIAM International Conference on Data Mining*, pages 270–278. SIAM, 2018.

[9] Jiaxi Tang and Ke Wang. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proc. of the 24th SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[10] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proc. of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016.

[11] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, pages 598–617. IEEE, 2016.

[12] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the International Conference on Machine Learning*, pages 3145–3153. JMLR. org, 2017.

[13] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

[14] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[15] Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. L-shapley and c-shapley: Efficient model interpretation for structured data. 2018.

[16] Marco Ancona, Cengiz Öztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for shapley values approximation. *arXiv preprint arXiv:1903.10992*, 2019.

[17] Igor Kononenko et al. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11(Jan):1–18, 2010.

[18] Pengtao Xie, Hongbao Zhang, Yichen Zhu, and Eric P Xing. Nonoverlap-promoting variable selection. In *Proceedings of the International Conference on Machine Learning*, 2018.

[19] Alfred M Bruckstein, David L Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.

[20] Emmanuel J Candes et al. Compressive sampling. In *Proc. of the international congress of mathematicians*, 2006.

[21] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[23] Wei Gao, Rong Jin, Shenghuo Zhu, and Zhi-Hua Zhou. One-pass auc optimization. In *Proceedings of the International Conference on Machine Learning*, pages 906–914, 2013.

[24] Majdi Khalid, Indrakshi Ray, and Hamidreza Chitsaz. Confidence-weighted bipartite ranking. In *International Conference on Advanced Data Mining and Applications*, pages 35–49. Springer, 2016.

[25] Mengting Zhan, Shilei Cao, Buyue Qian, Shiyu Chang, and Jishang Wei. Low-rank sparse feature selection for patient similarity learning. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 1335–1340. IEEE, 2016.

[26] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the International Conference on Machine Learning*, 2017.

[27] Felix Grün, Christian Rupprecht, Nassir Navab, and Federico Tombari. A taxonomy and library for visualizing learned features in convolutional neural networks. *arXiv preprint arXiv:1606.07757*, 2016.

[28] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proc. of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[29] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[30] Xingyu Gao, Steven CH Hoi, Yongdong Zhang, Ji Wan, and Jintao Li. Soml: Sparse online metric learning with application to image retrieval. In *Twenty-eighth AAAI conference on artificial intelligence*, 2014.

[31] Yiming Ying, Kaizhu Huang, and Colin Campbell. Sparse metric learning via smooth optimization. In *Advances in neural information processing systems*, pages 2214–2222, 2009.

[32] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

[33] Mengdi Huai, Chenglin Miao, Yaliang Li, Qiuling Suo, Lu Su, and Aidong Zhang. Metric learning from probabilistic labels. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1541–1550, 2018.

[34] Zhiwu Huang, Ruiping Wang, Shiguang Shan, Xianqiu Li, and Xilin Chen. Log-euclidean metric learning on symmetric positive definite manifold with application to image set classification. In *Proceedings of International Conference on Machine Learning*, pages 720–729, 2015.

[35] Joseph St Amand and Jun Huan. Sparse compositional local metric learning. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1097–1104, 2017.

[36] Pourya Zadeh, Reshad Hosseini, and Suvrit Sra. Geometric mean metric learning. In *Proceedings of International Conference on Machine Learning*, pages 2464–2471, 2016.

[37] Chen Huang, Chen Change Loy, and Xiaoou Tang. Local similarity-aware deep feature embedding. In *Advances in Neural Information Processing Systems*, pages 1262–1270, 2016.

[38] Marc T Law, Raquel Urtasun, and Richard S Zemel. Deep spectral clustering learning. In *Proceedings of International Conference on Machine Learning*, pages 1985–1994, 2017.

[39] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4004–4012, 2016.

[40] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. *arXiv preprint arXiv:1708.01682*, 2017.

[41] Jiazhi Ni, Jie Liu, Chenxin Zhang, Dan Ye, and Zhirou Ma. Fine-grained patient similarity measuring using deep metric learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1189–1198, 2017.

[42] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016.

[43] Qi Qian, Juhua Hu, Rong Jin, Jian Pei, and Shenghuo Zhu. Distance metric learning using dropout: a structured regularization approach. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery & data mining*, pages 323–332, 2014.

[44] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems*, pages 1988–1996, 2014.

[45] Xuezhe Ma, Yingkai Gao, Zhiting Hu, Yaoliang Yu, Yuntian Deng, and Eduard Hovy. Dropout with expectation-linear regularization. *arXiv preprint arXiv:1609.08017*, 2016.

[46] Ke Zhai and Huan Wang. Adaptive dropout with rademacher complexity regularization. In *Proceedings of ICLR*, 2018.

[47] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of International Conference on Machine Learning*, pages 1058–1066, 2013.

[48] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[49] Stéphan Clémençon, Gábor Lugosi, and Nicolas Vayatis. Ranking and scoring using empirical risk minimization. In *International conference on computational learning theory*, pages 1–15, 2005.

[50] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[51] Blake Mason, Lalit Jain, and Robert Nowak. Learning low-dimensional metrics. In *Advances in Neural Information Processing Systems*, pages 4139–4147, 2017.

[52] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[53] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, 2015.

[54] Henry Gouk, Bernhard Pfahringer, and Michael Cree. Fast metric learning for deep neural networks. *arXiv preprint arXiv:1511.06442*, 2015.

[55] Rong Jin, Shijun Wang, and Yang Zhou. Regularized distance metric learning: Theory and algorithm. In *Advances in Neural Information Processing Systems*, pages 862–870, 2009.

[56] Qiong Cao, Zheng-Chu Guo, and Yiming Ying. Generalization bounds for metric and similarity learning. *Machine Learning*, 102(1):115–132, 2016.

[57] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *Proceedings of International Conference on Machine Learning*, 2018.

[58] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[59] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[60] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[61] Sam Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. *arXiv preprint arXiv:1711.00138*, 2017.

[62] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[63] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding dqns. In *International Conference on Machine Learning*, pages 1899–1908, 2016.

[64] Laurens Weitkamp, Elise van der Pol, and Zeynep Akata. Visual rationalizations in deep reinforcement learning for atari games. In *Benelux Conference on Artificial Intelligence*, pages 151–165. Springer, 2018.

[65] Liu Yuezhang, Ruohan Zhang, and Dana H Ballard. An initial attempt of combining visual selective attention with deep reinforcement learning. *arXiv preprint arXiv:1811.04407*, 2018.

[66] Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. *arXiv preprint arXiv:1912.05743*, 2019.

[67] Kaidi Xu, Sijia Liu, Pu Zhao, Pin-Yu Chen, Huan Zhang, Quanfu Fan, Deniz Erdogmus, Yanzhi Wang, and Xue Lin. Structured adversarial attack: Towards general implementation and better interpretability. *arXiv preprint arXiv:1808.01664*, 2018.

[68] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. Interpretable deep learning under fire. In *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[69] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[70] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[71] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pages 9505–9515, 2018.

[72] Rahul Iyer, Yuezhang Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia Sycara. Transparency and explanation in deep reinforcement learning neural networks. In *Proc. of the AAAI/ACM Conference on AI, Ethics, and Society*, 2018.

[73] Chenglin Miao, Qi Li, Houping Xiao, Wenjun Jiang, Mengdi Huai, and Lu Su. Towards data poisoning attacks in crowd sensing systems. In *Proc. of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 111–120, 2018.

[74] Chenglin Miao, Qi Li, Lu Su, Mengdi Huai, Wenjun Jiang, and Jing Gao. Attack under disguise: An intelligent data poisoning attack mechanism in crowdsourcing. In *Proc. of the 2018 World Wide Web Conference*, pages 13–22, 2018.

[75] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1778–1787, 2018.

[76] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[77] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[78] Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Targeted attacks on deep reinforcement learning agents through adversarial observations. *arXiv preprint arXiv:1905.12282*, 2019.

[79] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2040–2042, 2018.

[80] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017.

[81] Xinghua Qu, Zhu Sun, Pengfei Wei, Yew-Soon Ong, and Abhishek Gupta. Minimalistic attacks: How little it takes to fool a deep reinforcement learning policy. *arXiv preprint arXiv:1911.03849*, 2019.

[82] Jianwen Sun, Tianwei Zhang, Xiaofei Xie, Lei Ma, Yan Zheng, Kangjie Chen, and Yang Liu. Stealthy and efficient adversarial attacks against deep reinforcement learning. *arXiv preprint arXiv:2005.07099*, 2020.

[83] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

[84] Mengdi Huai, Di Wang, Chenglin Miao, and Aidong Zhang. Towards interpretation of pairwise learning. In *Thirty-fourth AAAI Conference on Artificial Intelligence*, 2020.

[85] Amirata Ghorbani, James Wexler, James Zou, and Been Kim. Towards automatic concept-based explanations. *Advances in Neural Information Processing Systems*, 2019.

[86] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *Proceedings of the International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[87] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proceedings of the International Conference on Machine Learning*, pages 5338–5348. PMLR, 2020.

[88] Tejaswini Pedapati, Avinash Balakrishnan, Karthikeyan Shanmugam, and Amit Dhurandhar. Learning global transparent models consistent with local contrastive explanations. *Advances in Neural Information Processing Systems*, 33, 2020.

[89] Jeya Vikranth Jeyakumar, Joseph Noor, Yu-Hsi Cheng, Luis Garcia, and Mani Srivastava. How can i explain this to you? an empirical study of deep neural network explanation methods. *Advances in Neural Information Processing Systems*, 2020.

[90] Tom Heskes, Evi Sijben, Ioan Gabriel Bucur, and Tom Claassen. Causal shapley values: Exploiting causal knowledge to explain individual predictions of complex models. *arXiv preprint arXiv:2011.01625*, 2020.

[91] Matthew O'Shaughnessy, Gregory Canal, Marissa Connor, Mark Davenport, and Christopher Rozell. Generative causal explanations of black-box classifiers. *Advances in Neural Information Processing Systems*, 2020.

[92] Liuyi Yao, Yaliang Li, Sheng Li, Mengdi Huai, Jing Gao, and Aidong Zhang. Sci: Subspace learning based counterfactual inference for individual treatment effect estimation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3583–3587, 2021.

[93] Mengdi Huai, Di Wang, Chenglin Miao, and Aidong Zhang. Towards interpretation of pairwise learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4166–4173, 2020.

[94] Mengdi Huai, Chenglin Miao, Jinduo Liu, Di Wang, Jingyuan Chou, and Aidong Zhang. Global interpretation for patient similarity learning. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 589–594. IEEE, 2020.

[95] Chih-Kuan Yeh, Been Kim, Sercan O Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. *arXiv preprint arXiv:1910.07969*, 2019.

[96] Runjin Chen, Hao Chen, Jie Ren, Ge Huang, and Quanshi Zhang. Explaining neural networks semantically and quantitatively. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9187–9196, 2019.

[97] Yash Goyal, Amir Feder, Uri Shalit, and Been Kim. Explaining classifiers with causal concept effect (cace). *arXiv preprint arXiv:1907.07165*, 2019.

[98] Weibin Wu, Yuxin Su, Xixian Chen, Shenglin Zhao, Irwin King, Michael R Lyu, and Yu-Wing Tai. Towards global explanations of convolutional neural networks with concept attribution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8652–8661, 2020.

[99] Diana Mincu, Eric Loreaux, Shaobo Hou, Sebastien Baur, Ivan Protsyuk, Martin Seneviratne, Anne Mottram, Nenad Tomasev, Alan Karthikesalingam, and Jessica Schrouff. Concept-based model explanations for electronic health records. In *Proceedings of the Conference on Health, Inference, and Learning*, pages 36–46, 2021.

[100] Ann-Kathrin Dombrowski, Maximilian Alber, Christopher J Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *arXiv preprint arXiv:1906.07983*, 2019.

[101] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. Interpretable deep learning under fire. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

[102] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[103] Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Sai Suggala, David I Inouye, and Pradeep Ravikumar. On the (in) fidelity and sensitivity for explanations. *arXiv preprint arXiv:1901.09392*, 2019.

[104] Himabindu Lakkaraju, Nino Arsov, and Osbert Bastani. Robust and stable black box explanations. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2020.

[105] Puneet Mangla, Vedant Singh, and Vineeth N Balasubramanian. On saliency maps and adversarial robustness. *arXiv preprint arXiv:2006.07828*, 2020.

[106] Rahul Soni, Naresh Shah, Chua Tat Seng, and Jimmy D Moore. Adversarial tcav–robust and effective interpretation of intermediate layers in neural networks. *arXiv preprint arXiv:2002.03549*, 2020.

[107] Adam Ivankay, Ivan Girardi, Chiara Marchiori, and Pascal Frossard. Far: A general framework for attributional robustness. *arXiv preprint arXiv:2010.07393*, 2020.

[108] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. *arXiv preprint arXiv:1806.07538*, 2018.

[109] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[110] Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.

[111] Teruo Sunaga. Theory of an interval algebra and its application to numerical analysis. *RAAG memoirs*, 2(29-46):209, 1958.

[112] Mengdi Huai, Jianhui Sun, Renqin Cai, Liuyi Yao, and Aidong Zhang. Malicious attacks against deep reinforcement learning interpretations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 472–482, 2020.

[113] Anindya Sarkar, Anirban Sarkar, and Vineeth N Balasubramanian. Enhanced regularizers for attributional robustness. *arXiv preprint arXiv:2012.14395*, 2020.

[114] Alexandros Stergiou. The mind's eye: Visualizing class-agnostic features of cnns. *arXiv preprint arXiv:2101.12447*, 2021.

[115] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[116] Dmitry Kazhdan, Botty Dimanov, Mateja Jamnik, Pietro Liò, and Adrian Weller. Now you see me (cme): Concept-based model extraction. *arXiv preprint arXiv:2010.13233*, 2020.

[117] Max Losch, Mario Fritz, and Bernt Schiele. Interpretability beyond classification output: Semantic bottleneck networks. *arXiv preprint arXiv:1907.10882*, 2019.

[118] Lemeng Wu, Dilin Wang, and Qiang Liu. Splitting steepest descent for growing neural architectures. *Advances in Neural Information Processing Systems*, 32:10656–10666, 2019.

[119] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[120] Jiefeng Chen, Xi Wu, Vaibhav Rastogi, Yingyu Liang, and Somesh Jha. Robust attribution regularization. *arXiv preprint arXiv:1905.09957*, 2019.

[121] Yang Lu, Wenbo Guo, Xinyu Xing, and William Stafford Noble. Robust decoy-enhanced saliency maps. *arXiv preprint arXiv:2002.00526*, 2020.

[122] Zhengze Zhou, Giles Hooker, and Fei Wang. S-lime: Stabilized-lime for model explanation. *arXiv preprint arXiv:2106.07875*, 2021.

[123] Zekun Zhang and Tianfu Wu. Learning ordered top-k adversarial attacks via adversarial distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 776–777, 2020.

[124] Nurislam Tursynbek, Aleksandr Petiushko, and Ivan Oseledets. Geometry-inspired top-k adversarial perturbations. *arXiv preprint arXiv:2006.15669*, 2020.

[125] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[126] Alexander Levine, Sahil Singla, and Soheil Feizi. Certifiably robust interpretation in deep learning. *arXiv preprint arXiv:1905.12105*, 2019.

[127] Jimeng Sun, Fei Wang, Jianying Hu, and Shahram Edabollahi. Supervised patient similarity measure of heterogeneous patient records. *ACM SIGKDD Explorations Newsletter*, 14(1), 2012.

[128] Peilin Zhao, Steven CH Hoi, Rong Jin, and Tianbao Yang. Online auc maximization. In *Proceedings of the International Conference on Machine Learning*, pages 233–240, 2011.

[129] Purushottam Kar, Bharath Sriperumbudur, Prateek Jain, and Harish Karnick. On the generalization ability of online learning algorithms for pairwise loss functions. In *Proceedings of the International Conference on Machine Learning*, pages 441–449, 2013.

[130] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*.

[131] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy*, 2017.

[132] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 2006.

[133] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, 2009.

[134] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, 2014.

[135] Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially private online learning. In *Conference on Learning Theory*, 2012.

[136] Di Wang, Changyou Chen, and Jinhui Xu. Differentially private empirical risk minimization with non-convex loss functions. In *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[137] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 2011.

[138] Di Wang, Minwei Ye, and Jinhui Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems*, 2017.

[139] Abhradeep Guha Thakurta and Adam Smith. (nearly) optimal algorithms for private online learning in full-information and bandit settings. In *Advances in Neural Information Processing Systems*, pages 2733–2741, 2013.

[140] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of cryptography conference*, pages 635–658. Springer, 2016.

[141] Jaewoo Lee and Daniel Kifer. Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[142] Di Wang and Jinhui Xu. Differentially private empirical risk minimization with smooth non-convex loss functions: A non-stationary view. *Thirty-Third AAAI Conference on Artificial Intelligence, (AAAI-19), Honolulu, Hawaii, USA, January 27-February 1, 2019*, 2019.

[143] Stéphan Clémençon, Gábor Lugosi, Nicolas Vayatis, et al. Ranking and empirical minimization of u-statistics. *The Annals of Statistics*, 2008.

[144] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003.

[145] Brian Kulis and Peter L Bartlett. Implicit online learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 575–582, 2010.

[146] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60, 2010.

[147] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[148] Chenglin Miao, Wenjun Jiang, Lu Su, Yaliang Li, Suxin Guo, Zhan Qin, Houping Xiao, Jing Gao, and Kui Ren. Cloud-enabled privacy-preserving truth discovery in crowd sensing systems. In *Proceedings of SenSys*, pages 183–196, 2015.

[149] Xinghua Shi and Xintao Wu. An overview of human genetic privacy. *Annals of the New York Academy of Sciences*, 1387(1):61–72, 2017.

[150] Chenglin Miao, Lu Su, Wenjun Jiang, Yaliang Li, and Miaomiao Tian. A lightweight privacy-preserving truth discovery framework for mobile crowd sensing systems. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[151] Wei Feng, Zheng Yan, Hengrun Zhang, Kai Zeng, Yu Xiao, and Y Thomas Hou. A survey on security, privacy, and trust in mobile crowdsourcing. *IEEE Internet of Things Journal*, 5(4):2971–2992, 2017.

[152] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284, 2006.

[153] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[154] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1187–1198, 2014.

[155] Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han. A confidence-aware approach for truth discovery on long-tail data. *Proceedings of the VLDB Endowment*, 8(4):425–436, 2014.

[156] Yaliang Li, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. Conflicts to harmony: A framework for resolving conflicts in heterogeneous data by truth discovery. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 28(8):1986–1999, 2016.

[157] Chenglin Miao, Wenjun Jiang, Lu Su, Yaliang Li, Suxin Guo, Zhan Qin, Houping Xiao, Jing Gao, and Kui Ren. Privacy-preserving truth discovery in crowd sensing systems. *ACM Transactions on Sensor Networks (TOSN)*, 15(1):9, 2019.

[158] Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust statistics: The approach based on influence functions*. Wiley Online Library, 2011.

[159] Johannes Gehrke, Michael Hay, Edward Lui, and Rafael Pass. Crowd-blending privacy. In *Annual Cryptology Conference*, pages 479–496. 2012.

[160] Vincent Bindschaedler, Reza Shokri, and Carl A Gunter. Plausible deniability for privacy-preserving data synthesis. *Proceedings of the VLDB Endowment*, 10(5):481–492, 2017.

[161] Jeff Pasternack and Dan Roth. Knowing what to believe (when you already know something). In *Proceedings of Coling*, pages 877–885, 2010.

[162] Mengting Wan, Xiangyu Chen, Lance Kaplan, Jiawei Han, Jing Gao, and Bo Zhao. From truth discovery to trustworthy opinion discovery: An uncertainty-aware quantitative modeling approach. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1885–1894, 2016.

[163] Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava. Truth finding on the deep web: Is the problem solved? *Proceedings of the VLDB Endowment*, 6(2):97–108, 2012.

[164] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 123–134, 2010.

[165] Moritz Hardt and Guy N Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st annual symposium on foundations of computer science*, pages 61–70, 2010.

[166] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 765–774, 2010.

[167] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84, 2007.

[168] Vincent Bindschaedler and Reza Shokri. Synthesizing plausible privacy-preserving location traces. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 546–563, 2016.

[169] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)*, 60(2):12, 2013.

[170] Di Wang, Marco Gaboardi, and Jinhui Xu. Empirical risk minimization in non-interactive local differential privacy revisited. In *Advances in Neural Information Processing Systems*, pages 965–974, 2018.

[171] Di Wang, Adam Smith, and Jinhui Xu. Noninteractive locally private learning of linear models via polynomial approximations. In *Algorithmic Learning Theory*, pages 897–902, 2019.

[172] Holger Rauhut. Compressive sensing and structured random matrices. *Theoretical foundations and numerical methods for sparse recovery*, 9:1–92, 2010.

[173] Colin McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.

[174] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Mach. Learn.*, 2002.

[175] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 2007.