Robust Machine Learning Methods and Applications

by

Shu Liu

A Dissertation Submitted in Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy in Computational and Data Science

Middle Tennessee State University

May 2023

Dissertation Committee:

Dr. Qiang Wu, Chair

Dr. Abdul Khaliq

Dr. Cen Li

Dr. Joshua L. Phillips

Dr. Xin Yang

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my advisor, Dr. Qiang Wu, for the continuous support and encouragement on my Ph.D study and research. His patience and immense knowledge, especially the spirit of exploring new technologies, have benefited me a lot. I am thankful to Dr. Donglin Wang and Dr. Wandi Ding for their support and advice on my research. I would like to thank Dr. John F. Wallin for his guidance and support during my Ph.D study. I would like to thank Dr. Abdul Khaliq and Dr. Joshua L. Phillips for their patient teaching in the doctoral program courses and their professional opinions as committee members. I would like to thank the rest of my dissertation committee members, Dr. Cen Li and Dr. Xin Yang, for their insightful comments and encouragement on my dissertation. I would like to thank Middle Tennessee State University and the Computational and Data Science Program for funding my graduate studies and research. My sincere thanks also go to all the professors who had taught me during my Ph.D study: Dr. Jing Kong, Dr. Ralph Butler, Dr. Yeqian Liu, Dr. Zachariah Sinkala, Dr. Ginger Rowell, Dr. Ping Zhang, Dr. Lu Xiong and Dr. Ramchandra Rimal, for their professional and conscientious teaching and support. Finally, I would like to thank my family and fiancée for providing unending support.

ABSTRACT

This dissertation introduces a variety of methods motivated from the use of robust losses in machine learning and deep neural networks and explores their applications. Topics include novel linear and nonlinear approaches for imbalanced data classification, their applications in functional magnetic resonance imaging, robust deep neural networks and the application in adversarial attacks. Robustness is mainly used to test whether the model can still maintain accuracy of judgment in the face of small changes in input data, that is, whether the performance of the model is stable in the face of certain changes. The level of robustness directly impacts the generalization ability of machine learning models.

In this dissertation, we have used different robust losses to propose, analyze, and evaluate models. First, robust support vector classifier loss was adapted to propose two new classifiers, the pairwise robust support vector machine and its kernel counterpart, which are shown more efficient for imbalanced data classification. Then, correntropy loss was used to propose two robust deep neural networks and their two-stage implementations. Finally, correntropy loss was introduced for adversarial attack to further analyze the robustness of an image classification model. Simulations and case studied show that with the introduction of robust losses into machine learning and deep learning, the newly proposed models have better performance than traditional models and can improve model robustness.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Robustness exists widely in many fields, and it acquires different meanings when combined with actual situations in different fields. The concept of robustness first appeared in statistics and was used to evaluate the ability of a statistical method to maintain effectiveness in the condition that the assumptions are violated, for example, the training data contains outliers [14] or the model is under attack [79]. Since the 1970s, robustness has been gradually introduced into the study of control theory [19] to represent the insensitivity of the control system to characteristic or parameter disturbances. In the field of biology, biological robustness is best reflected in the adaptation of organisms to the environment [3], and is a ubiquitous feature in biological systems. In this dissertation, we will discuss the performance of robustness from two aspects of traditional machine learning and neural networks.

1.1 Robustness in machine learning

One of the earliest works in the field of robust statistics was the concept of robustness proposed by Huber [49], who also provided a framework for developing robust statistical methods. In the 1990s, robustness became a research topic in the field of machine learning. Scholars began to explore machine learning algorithms that are more robust to data changes, such as outliers, missing data, and noise.

Huber [50] elaborated on robust estimation in his early published book. He categorized statistical estimation methods into M-estimators, R-estimators and L-estimators. Among them, M-estimators are the most commonly used ones in robust regression [42]. Both least squares estimation and maximum likelihood estimation

are M estimation methods. Based on M-estimators, Huber proposed Huber Loss [50]

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta, \\ \delta\,|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise,} \end{cases}$$

which uses squared error for small residuals and absolute error for large residuals. Huber loss is less sensitive to outliers in the data than squared error loss. At value 0, it is also differentiable. It's basically an absolute value that becomes a squared value when the error is small. In fact, the size of the loss value depends on a hyperparameter δ. When $\delta \sim 0$, Huber loss tends to mean absolute error, when $\delta \sim \infty$, Huber loss tends to mean squared error. Huber Loss is still used in a large number of robust machine learning algorithms up to now [35, 41, 71].

Since outlier detection [82] has been widely studied in robust regression, scholars have also been working on the robustness of algorithms in classification problems. Due to the popularity of support vector machines (SVM), many robustness research results based on SVM have been produced. Xu et al. [117] in 2009 studied the equivalence between support vector machines and the robust formulation of hinge loss classifiers. Shen et al. [94] proposed a ψ-learning method using a truncated set of non-convex loss functions and demonstrated its superiority over SVM on the binary classification problem of breast cancer data. Huang et al. [48] proposed an SVM classifier with pinball loss that is noise insensitive and resampling stable. In 2016, Feng et al. [28] introduced a robust and smooth family of classification loss functions into the SVM algorithm and proposed robust support vector machines for classification (RSVC) which showed clear advantages on datasets with 20% labels flipped. In the discussion on the performance of robustness in traditional learning methods, our research is carried out on the basis of the RSVC algorithm. We introduce pairwise RSVC loss

for imbalanced classification tasks and analyze its performance.

1.2 Robustness in deep neural networks

With the continuous improvement of computing power and algorithms, deep neural networks (DNN) have become an essential tool in the field of artificial intelligence and it is currently the basis of many artificial intelligence applications. Its outstanding performance stems from its ability to use statistical learning methods to extract high-level features from raw sensory data and obtain effective representations of the input space from a large amount of data. At present, DNN has been applied in many fields such as autonomous driving [103], cancer detection [22], video games [89] and education [122].

Neural network theory is still a black box for many people. In order to make it better applied in some fields involving life safety or physical and mental health, we must emphasize its robustness. At present, many studies have found that neural networks are very sensitive to small disturbances [116]. This means that for neural networks, human-made attacks will be more difficult to detect and defend against.

One direction of recent research on improving neural networks has mainly focused on defense methods. An idea is to try to learn DNNs that are provably robust against perturbations with bounded norms. For example, in 2018, Weng et al. [113] transformed neural network robustness problem to the estimation problem of the local Lipschitz constant and proposed a network robustness metric using the method of extreme value theory. This method can evaluate unknown attack methods. Wong and Kolter [114] proposed a ReLU-based deep classifier for classification and showed that the classifier can be robust to norm-bounded adversarial perturbations on the training data. Another idea is to incorporate adversarial data into the training process

to make the trained network robust to attacks; see for example [15, 78]. For the robustness in deep neural networks tasks, we use the correntropy loss [25] which has been proved effective to handle non-Gaussian noises. We first use correntropy loss to train the deep neural network and analyze its performance. Then we introduce correntropy loss into the adversarial attack to check the classification accuracy of the adversarial examples generated by correntropy loss in the networks.

1.3 Outline of dissertation

In this chapter, we have overviewed the background of robust machine learning. The rest of this dissertation is organized as follows.

In Chapter 2 we propose pairwise robust support vector machine (PRSVM). In Chapter 3, we apply PRSVM to the classification and analysis of the autism brain imaging data exchange (ABIDE). In Chapter 4 we propose nonlinear pairwise robust support vector machine using the kernel trick.

In Chapter 5, we introduce correntropy loss into deep neural networks and propose two-stage algorithms for deep neural networks. In Chapter 6, we introduce correntropy loss into adversarial attacks and analyze the classification performance of adversarial examples against image classification models.

We end with Chapter 7 by a summary of this dissertation and some discussions of potential future research topics that are related to the work of this dissertation.

CHAPTER 2

PAIRWISE ROBUST SUPPORT VECTOR MACHINE

2.1 Overview

In this chapter we propose a novel method called pairwise robust support vector machine (PRSVM) to overcome the difficulty of imbalanced data classification. It adapts the non-convex robust support vector classification loss to the pairwise learning setting. In the training process, samples from the minority class and the majority class always appear as pairs. This automatically balances the impact of two classes. Simulations and real-world applications show that PRSVM is highly effective.

2.2 Introduction

Imbalanced data appear ubiquitously in real-word applications. For instance, in computer-aided medical diagnosis patients with certain concerned disease such as cancer or diabetes count only a very small fraction of the screening population [20, 69, 80, 97]. For spam detection system, the number of spam messages is usually far less than useful ones [23]. Intrusion detection requires detecting malicious and unauthorized activities that attack computer systems. Although rapid increase of such attacks has been seen along the popularization of computers, it is believed they are still "outliers" compared to normal activities [102].

Classification of imbalanced data is a challenging task. Approaches that are commonly used and effective on balanced data, such as support vector machines and logistic regression, could perform poorly on imbalanced data. One could get small total classification error as long as they classify data of the majority class correctly.

The accuracy of classification of the minority class is usually low though more often it is the concerned class. Some reasons could be the absolute scarcity of minority samples [112], an inappropriate evaluation method [77], or the high degree of overlap of minority with the majority [73].

Researchers have been paying increasing attention to imbalanced data classification in the past two decades. The most common approach is to preprocess the data via resampling techniques before training. The oversampling approach resamples or duplicates the minority class so that its sample size increases to be comparable with majority class [7,32,56,110] while the undersampling approach downsamples the majority class to have size comparable with the minority one [32,56,57,65]. Training data becomes balanced after preprocessing and therefore traditional classification methods such as support vector machines and AdaBoost can be efficiently used. However, these preprocessing tricks have some known drawbacks. The oversampling method may increase the likelihood of overfitting due to duplicated data. On the contrary, the undersampling method may cause loss of information because some useful data present in the majority class might be eliminated. Moreover, when the minority class is too small, the undersampling method will generate an undersized training set, which may decrease the performance of classifiers. Menardi and Torelli [70] proposed random over-sampling examples (ROSE) by a smoothed bootstrap-based technique. It simultaneously oversamples the minority class while undersampling the majority class and more often shows slightly better performance by reducing information loss and overfitting. In addition to these preprocessing tricks, researchers had also considered to modify classification algorithms or data characterizations to handle imbalanced data. For instance, Veropoulos et al. [106] proposed to adjust the error costs of different classes in the training of support vector machines, which places a large error

cost on the minority class and a small error cost on the majority class to balance the impact of skewed sample sizes. Later Akbani et al. [1] proposed to combine the different error costs idea from [106] with synthetic minority oversampling technique from [104]. In [39] appropriate feature selection was combined with naive Bayes to handle imbalanced text data.

In this chapter we propose a new imbalanced data classification approach in the pairwise learning framework. Pairwise learning arises naturally from metric learning, ranking, and information theoretic learning. When applied to binary classification problems, observations from the minority class and the majority class always appear as pairs for the loss evaluation. This automatically balances the impact of the two classes regardless of their sizes. It avoids the drawbacks of resampling techniques and therefore is potentially superior.

2.3 Traditional machine learning methods

Many classification methods have been proposed in the literature of machine learning resaerch. Here we review several of them that will be used in this dissertation.

2.3.1 Support vector machine

Support vector machines (SVMs) are classification models. The most basic SVM is a linear binary classifier with the largest margin defined in the feature space. SVM also has the kernel version, which makes it an essentially non-linear method. The learning strategy of SVM is to maximize the margin between the data points representing two classes, which can be formalized as a problem solvable using convex quadratic programming. It is also equivalent to the problem of minimizing the regu-

larized hinge loss function. The basic idea of SVM learning is to find the separating hyperplane $w^\top x + b = 0$ which could correctly partition the training dataset and has the largest geometric margin. For linearly separable datasets, there are infinitely many such hyperplanes, but the separating hyperplane with the largest geometric margin is unique.

For a given training dataset $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ on a character space, assume that $x_i \in \mathbb{R}^n$ is the ith observation and $y_i \in \{+1, -1\}$ is the class label. We say y_i admits a positive label if $y_i = +1$ and a negative label if $y_i = -1$ Assume that the training dataset is linearly separable, that is, there exists $w \in \mathbb{R}^n$ such that the hyperplane $w^\top x + b = 0$ separates the two classes in the sense that $w\top x_i + b > 0$ for $y_i = +1$ and $w^\top x_i + b < 0$ for $y_i = -1$. We define the geometric margin of the hyperplane with respect to the sample points (x_i, y_i) as $\gamma_i = y_i \left(\frac{w}{\|w\|} \cdot x_i + \frac{b}{\|w\|} \right)$, which is the distance of the observation x_i to the hyperplane. The minimum geometric margin of the hyperplane with respect to all sample points is $\gamma = \min\limits_{i=1,2,...,N} \gamma_i$. According to the above definition, the solution of the maximum margin problem of the SVM model can be expressed as the following constrained optimization problem:

$$\max_{w,b} \gamma$$
$$s.t. \quad y_i \left(\frac{w}{\|w\|} \cdot x_i + \frac{b}{\|w\|} \right) \geq \gamma, \quad i = 1, 2, ..., N.$$

Dividing both sides of the constraint by γ, we will get

$$y_i \left(\frac{w}{\|w\|\gamma} \cdot x_i + \frac{b}{\|w\|\gamma} \right) \geq 1.$$

Note $\|w\|$ and γ are both scalars. Rewrite $\frac{w}{\|w\|\gamma}$ as w and $\frac{b}{\|w\|\gamma}$ as b, the minimum geometric margin becomes $\frac{1}{\|w\|}$ and the constraint becomes

$$y_i (w \cdot x_i + b) \geq 1, \quad i = 1, 2, ..., N. \tag{2.1}$$

Therefore, the optimization problem can be reformulated to maximize $\frac{1}{\|w\|}$ under the constraint 2.1. Note maximizing $\frac{1}{\|w\|}$ is also equivalent to minimizing $\frac{1}{2}\|w\|^2$. So the SVM model can be expressed as the following constrained optimization problem:

$$\min \frac{1}{2}\|w\|^2$$

$$s.t. \quad y_i\left(w^\top x_i + b\right) \geq 1, \quad i = 1, 2, ..., N$$

2.3.2 AdaBoost

AdaBoost is the abbreviation of "Adaptive Boosting". Its self-adaptation lies in that the weight of a sample that was misclassified by the previous basic classifier will increase, while the weight of a correctly classified sample will decrease and be used again for training the next base classifier. At the same time, in each round of iterations, a new weak classifier is added, and the final strong classifier is not determined until a predetermined small error rate is reached or a pre-specified maximum number of iterations is reached.

For the given data set $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where $x_i \in \mathbb{R}^n$, $y_i \in \{+1, -1\}, i = 1, 2, ..., N$. AdaBoost first initializes the weight distribution of the training data. Usually each sample is given the same weight $w_i = \frac{1}{N}$ at the beginning. The next step is training the weak classifier $h_i : X \to \{+1, -1\}$. The specific training process is such that if a certain training sample point is accurately classified by the weak classifier h_i, then in the construction of the next training set, its corresponding weight should be reduced. On the contrary, if a certain training sample point is misclassified, then its weight should increase. The sample set with updated weights is used to train the next classifier, and the whole training process is carried out

iteratively. Finally, we combine the trained weak classifiers into a strong classifier

$$H = \text{sign}\left(\sum_{i=1}^{I} \alpha_i h_i(x)\right)$$

where I represents the number of iterations and α_i's represent the weight of the weak classifiers.

2.3.3 Naive Bayes

Naive Bayes Classification is based on Bayes' theorem which states

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

where A, B represent two events, $P(A), P(B)$ represent the probability of the two events, $P(A|B)$ and $P(B|A)$ indicate the conditional probability of event A occurring under the premise that event B has already occurred and the conditional probability of event B occurring under the premise that event A has already occurred, respectively. This theorem solves the problems often encountered in real life: given a certain conditional probability, how one obtains the probability of two events exchanged.

The ideological basis of Naive Bayes is to solve the maximum probability of occurrence of each category under the condition of the occurrence of the given item to be classified. Let $x = \{a_1, a_2, ..., a_m\}$ be an item to be classified, for each a_j, $j = 1, 2, ..., m$ is a feature attribute of x, $C = \{y_1, y_2, ..., y_n\}$ is the category set. Then statistically obtain the conditional probability estimation of each feature attribute under each category $P(a_j|y_i)$, where $j = 1, 2, ..., m$ and $i = 1, 2, ..., n$. If each feature attribute is conditionally independent, then according to Bayes' theorem, we will get

$$P(y_i|x) = \frac{P(x|y_i)P(y_i)}{P(x)}$$

Since the denominator is constant for all classes, only the largest numerator is required. Computing

$$\begin{aligned} P(x|y_i)P(y_i) &= P(a_1|y_i)P(a_2|y_i)...P(a_m|y_i)P(y_i) \\ &= P(y_i)\prod_{j=1}^{m} P(a_j|y_i), \quad i = 1, 2, ..., n, \end{aligned}$$

if

$$P(x|y_K)P(y_K) = max\{P(x|y_1)P(y_1), P(x|y_2)P(y_2), ..., P(x|y_n)P(y_n)\},$$

then

$$x \in y_K.$$

2.4 Resampling methods for imbalanced classification

When we use $y = w^\top x + b$ to classify the sample x, we are actually comparing $p = \frac{e^y}{1+e^y}$ with a threshold. For example, usually when $y > 0$ and $p > 0.5$, x will be judged as a positive example, otherwise it is a negative example. p actually represents the possibility of a positive example and $\frac{p}{1-p}$ represents the ratio of the possibility of a positive example to the possibility of a negative example. The decision rule of the classifier is such that if $\frac{p}{1-p} > 1$, this sample is predicted to be positive.

A threshold of 0.5 indicates that the classifier considers the true ratio of positives to negatives to be 1:1. However, when the sample categories in the training set are imbalanced, let m^+ represent the number of positive examples, and m^- represent the number of negative examples, the observed ratio is $\frac{m^+}{m^-}$. Since it is usually assumed that the training set is an unbiased sampling of the real population, the observed ratio approximates the real ratio. As long as the predicted ratio of the classifier is higher than the observed ratio, it should be judged as a positive example, that is if

$\frac{p}{1-p} > \frac{m^+}{m^-}$, this sample is predicted to be positive. But when the threshold is 0.5, the classifier makes decisions based on balanced sample categories, so the predicted value needs to be adjusted. We can let

$$\frac{p'}{1-p'} = \frac{p}{1-p}\frac{m^-}{m^+}.$$

This is the rescaling strategy in imbalanced class learning, but in real cases, the assumption that "the training set is an unbiased sampling of the real population" is often not true. We cannot effectively infer the true probability based on the observed probability of the training set, so we cannot directly apply the rescaling strategy. At present, the method of resampling is usually used.

The undersampling (also known as downsampling) method "undersamples" the negative examples in the training set, removes some negative examples, makes the number of positive and negative examples close, and then performs learning.

In contrast, the oversampling (also known as upsampling) method "oversamples" the positive examples in the training set, adding some positive examples so that the number of positive and negative examples is close, and then performs learning.

Random over sampling examples (ROSE) is also a method for generating artificial data. The ROSE method creates an artificial data sample by enlarging the feature space of the minority and majority classes. New samples are obtained from conditional kernel density estimates for these two classes, and data generated using ROSE are believed to better estimate the original data.

2.5 Pairwise robust support vector machine

Given a data of n observations $(x_i, y_i), i = 1, \ldots, n$, with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$, the classification error of a real-valued classifier $f : \mathbb{R}^p \to \mathbb{R}$ is defined by the 0-1 loss:

$$\sum_{i=1}^{n} L_{0\text{-}1}(y_i f(x_i)) = \sum_{i=1}^{n} \mathbb{I}_{\{y_i f(x_i) < 0\}},$$

where $\mathbb{I}_{(\cdot)}$ is the indicator function taking values 1 when the condition is true and 0 otherwise. The optimization of classification error is known to be NP hard due to discontinuity of the 0-1 loss. The success of large margin classifiers such as support vector machines and AdaBoost lies on the use of surrogate loss. In support vector machines, hinge loss is used:

$$L_{\text{hinge}}(yf(x)) = (1 - yf(x))_+ = \max(0, 1 - yf(x)).$$

Support vector machines were extensively studied and successfully used in numerous fields; see e.g. [10, 87, 100] and references therein. Though, support vector machines could become less robust as outliers present. Among various robustification efforts, Feng et al. [28] proposed robust support vector classifier (RSVC), which uses the smooth non-convex surrogate loss

$$L_{\text{RSVC}}(yf(x)) = \sigma^2 \left(1 - \exp\left(\frac{(1 - yf(x))_+^2}{\sigma^2}\right)\right)$$

to replace the hingle loss and handles outliers well, where σ is a tunable parameter. See Fig. 2.1 for a comparison of the three loss functions.

Our approach to handle imbalanced data classification is motivated by using RSVC loss in the pairwise learning framework. A good real-valued classifier f should produce $\text{sign}(f(x_i)) = y_i$ as many as possible. Consequently, for each pair of observations (x_i, y_i) and (x_j, y_j), if $y_i = 1$ and $y_j = -1$, we would expect $f(x_i) - f(x_j) > 0$;

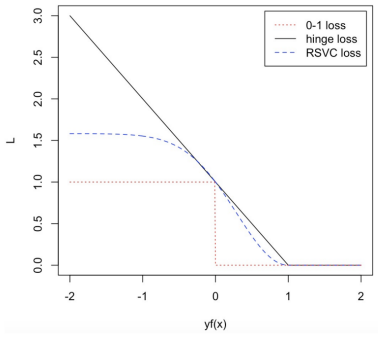


Figure 2.1. 0-1 loss, hinge loss, and RSVC loss with $\sigma = 1$ (scaled so that all three losses have value 1 at $yf(x) = 0$ for comparison purpose.)

if $y_i = -1$ but $y_j = 1$, we expect $f(x_i) - f(x_j) < 0$; when $y_i = y_j$ we have no expectations. Using the notation

$$y_{ij} = \begin{cases} 1, & \text{if } y_i = 1 \text{ and } y_j = -1; \\ -1, & \text{if } y_i = -1 \text{ and } y_j = 1, \end{cases}$$

we should have $y_{ij}(f(x_i) - f(x_j)) > 0$. We adapt the RSVC loss and define the pairwise loss as

$$L(f, (x_i, y_i), (x_j, y_j)) = \sigma^2 \left(1 - \exp\left(\frac{(1 - y_{ij}(f(x_i) - f(x_j)))_+^2}{\sigma^2}\right)\right).$$

We name this loss function as the pairwise robust support vector classifier loss. It is calculated by pairing a sample from the minority class with a sample from the majority class, ensuring that both classes make equal contributions during model training.

In this chapter we focus on linear classifiers $f(x) = w^\top x + b$ with $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$. The optimization problem associated to the pairwise robust support vector

classifier is

$$\min_{w} \sum_{i=1}^{n} \sum_{j:y_j \neq y_i} \sigma^2 \left(1 - \exp\left(\frac{(1 - y_{ij} w^\top (x_i - x_j))_+^2}{\sigma^2}\right)\right). \tag{2.2}$$

This can be solved by either the gradient descent algorithm or iterative least square method. The gradient descent algorithm is shown in Algorithm 1 and Algorithm 2.

Note that the intercept b does not appear in (2.2) because it is canceled in calculation of the difference $f(x_i) - f(x_j) = w^\top (x_i - x_j)$. We need to figure out appropriate intercept b separately. This is done as follows: let $\hat{w}$ be the solution to the minimization problem (2.2). For $b \in \mathbb{R}$, let $\mathcal{E}_+(b)$ and $\mathcal{E}_-(b)$ denote the false positive rate (FPR) and false negative rate (FNR) of the classifier $\hat{w}^\top x + b$, respectively. Here FPR and FNR mean the number of true positive and true negative samples detected divided by the number of all true positive and true negative samples. Define

$$\mathcal{E}(b) = \max(\mathcal{E}_+(b), \mathcal{E}_-(b))$$

and the intercept is estimated by

$$\hat{b} = \min_{b \in \mathbb{R}} \mathcal{E}(b). \tag{2.3}$$

We call our method pairwise robust support vector machine (PRSVM). It has several advantages. First, the impact of the positive class and negative class is automatically balanced without need of resampling. This avoids overfitting or loss of information suffered by resampling techniques. Second, it inherits the robustness of the RSVC loss.

2.6 Simulations and applications

In this section, simulation studies and real-world applications are used to illustrate the effectiveness of PRSVM to handle imbalanced data. We will not only com-

Input: w, sample from positive class x_+, sample from negative class x_-, σ

Output: $L(f,(x_i,y_i),(x_j,y_j)),\ \frac{\partial L}{\partial w}$

let $loss = 0$;

let $\frac{\partial L}{\partial w} = [0,0,....,0]$;

for $x_i \in x_+$ **do**

  for $x_j \in x_-$ **do**

    compute $loss = loss + \sigma^2(1 - \exp(\frac{-\max(0,1-(w^\top x_i - w^\top x_j))^2}{\sigma^2}))$

    compute $\frac{\partial L}{\partial w} =$

    $\frac{\partial L}{\partial w} - 2\exp(\frac{-\max(0,1-(w^\top x_i - w^\top x_j))^2}{\sigma^2})(\max(0,1-(w^\top x_i - w^\top x_j)))(x_i - x_j)$;

  end

end

compute $L(f,(x_i,y_i),(x_j,y_j)) = loss/(N_{x_+} \times N_{x_-})$;

compute $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial w}/(N_{x_+} \times N_{x_-})$;

return $L(f,(x_i,y_i),(x_j,y_j)),\ \frac{\partial L}{\partial w}$

Algorithm 1: Compute $L(f,(x_i,y_i),(x_j,y_j))$ and $\frac{\partial L}{\partial w}$ for PRSVM

Input: Sample from positive class x_+, sample from negative class x_-, y, σ, step size s, max iter

Output: w

solve initial w through x and y ;

compute initial loss;

while $loss_{new} - loss_{old} < 0.01$ *or* $iter > maxiter$ *or* $\|\frac{\partial L}{\partial w}\| < 0.00001$ **do**

    let $loss_{old} = loss$;

    compute $\frac{\partial L}{\partial w}$;

    compute $w_{new} = w - s\frac{\partial L}{\partial w}$;

    compute $loss_{new}$ through w_{new} ;

    if $loss_{\text{new}} < loss$ **then**

        $loss = loss_{\text{new}}$;

    else

        $loss = loss_{\text{old}}$;

    end

end

Return w;

Algorithm 2: Compute w for PRSVM

pare it with three traditional classification methods: support vector machine (SVM), AdaBoost, and Naive Bayes (NB), but also compare with the three resampling preprocessing techniques: Undersampling (US), Oversampling (OS), and Random Over-Sampling Examples (ROSE).

PRSVM is not very sensitive to the choice of the parameter σ. A moderate choice usually give sufficiently good results. We selected $\sigma = 1$ in all experiments. All other methods are implemented in R with standard packages. In particular, for SVM we used the `train` function in `caret` package. For fair comparison, method `svmlinear` is used to produce linear classifiers. Other parameters are kept default. AdaBoost used the function `boosting` from `adabag` package. The number of iterations is set as 200. Naive Bayes used the function `naiveBayes` from `e1071` package with default parameters. The implementation of three resampling methods used the `ROSE` package. For undersampling and oversampling methods, the resampling sample size is set as twice of the amount of minority class or majority class, respectively, so that the resampling process stops when two classes have the same size. For ROSE all parameters are set as default. SVM linear classifier is used to classify data after the data is balanced.

For imbalanced data classification, overall accuracy does not make much sense, especially when the majority class dominates. We will look at the FPR and FNR simultaneously. A balanced FPR and FNR implies the minority class has been equally addressed. We also evaluate the area under the receiver operating characteristic (ROC) curve (AUC), which is widely accepted as a balanced accuracy metric for imbalanced data classification problems; see e.g. [47]. The larger AUC, the better.

2.6.1 Simulation studies

We assume the data come from $\mathbb{R}^2$. For the positive class, $x_i = (x_{i,1}, x_{i,2})$ has both features $x_{i,1}$ and $x_{i,2}$ sampled from normal distribution with mean 3 and variance 1. For the negative class, both features are sampled from normal distribution with mean 5 and variance 1. The positive class will be the minority class. In the experiment, we will first set an imbalance ratio, that is, the ratio between sample sizes of the positive class and the negative class, say 1:k with some $k > 1$. With training sample size n, we will generate $\frac{n}{k+1}$ samples for the positive class and $\frac{kn}{k+1}$ samples for the negative class. The performance of each classifier will then be evaluated on a test set which contains 1000 samples from the positive class and $1000k$ samples from the negative class.

To compare the performance of seven methods, we fix the number of training samples $n = 200$ and investigate the impact of imbalance ratio by varying it from 1:2 to 1:4 and then to 1:8. Each experiment is repeated 50 times. The average FPR, FNR and AUC with standard error for the seven classifiers are reported in Table 2.1. The results indicate that SVM, AdaBoost and Naive Bayes deteriorate fast as imbalance ratio increases. FPR drops at the price of increasing FNR and AUC decreases. The three resampling techniques and PRSVM still give balanced FPR and FNR and AUC drop is not significant. In all scenarios PRSVM achieves the largest AUC.

2.6.2 Application I: Diabetes data

Diabetes Data [20] (`https://datahub.io/machine-learning/diabetes`) is a multivariate data set with 8 attributes measuring patients' medical conditions. The response variable denotes whether a patient is tested positive (+1) or negative (−1) for

Table 2.1. Classification performance of seven classifiers on simulated data with $n = 200$ and varying imbalance ratios

1:k	Method	FNR	FPR	AUC
1:2	SVM	0.1284 (0.0004)	0.0471 (0.0002)	0.9123 (0.0001)
	ADABOOST	0.1652 (0.0006)	0.0726 (0.0003)	0.8811 (0.0002)
	NB	0.1267 (0.0003)	0.0475 (0.0001)	0.9129 (0.0001)
	US	0.0856 (0.0004)	0.0779 (0.0003)	0.9182 (0.0001)
	OS	0.0873 (0.0004)	0.0768 (0.0003)	0.9179 (0.0001)
	ROSE	0.0841 (0.0005)	0.0831 (0.0004)	0.9164 (0.0001)
	PRSVM	0.0850 (0.0004)	0.0781 (0.0004)	0.9767 (<0.0001)
1:4	SVM	0.1910 (0.0004)	0.0272 (0.0001)	0.8909 (0.0002)
	ADABOOST	0.2177 (0.0005)	0.0460 (0.0002)	0.8682 (0.0002)
	NB	0.1840 (0.0003)	0.0284 (<0.0001)	0.8938 (0.0001)
	US	0.0814 (0.0004)	0.0867 (0.0004)	0.9159 (0.0001)
	OS	0.0898 (0.0003)	0.0754 (0.0003)	0.9174 (<0.0001)
	ROSE	0.0801 (0.0003)	0.0848 (0.0003)	0.9176 (<0.0001)
	PRSVM	0.0876 (0.0004)	0.0780 (0.0004)	0.9771 (<0.0001)
1:8	SVM	0.2643 (0.0003)	0.0153 (<0.0001)	0.8602 (0.0002)
	ADABOOST	0.3122 (0.0006)	0.0283 (0.0001)	0.8298 (0.0003)
	NB	0.1267 (0.0003)	0.0475 (0.0001)	0.9129 (0.0001)
	US	0.0960 (0.0004)	0.0815 (0.0003)	0.9113 (0.0001)
	OS	0.0942 (0.0003)	0.0794 (0.0003)	0.9132 (<0.0001)
	ROSE	0.0891 (0.0005)	0.0803 (0.0004)	0.9153 (<0.0001)
	PRSVM	0.0968 (0.0004)	0.0729 (0.0003)	0.9766 (<0.0001)

diabetes. The data contains 268 positive cases and 500 negative cases, leading to an imbalance ratio close to 1:2.

We randomly sample 1/3 of positive cases and 1/3 of negative cases to form a training set so that the the original imbalance ratio is maintained in the training process. The remaining cases are used to test the performance of the seven classifiers. The experiments are repeated 50 times and the average FPR, FNR and AUC are reported Table 2.2. Similar to our simulation studies, SVM, AdaBoost and Naive Bayes has the poorest performance. The three resampling methods performs better. Although PRSVM has the total error (FPR+FNR) slightly larger than resampling methods, it has the best balanced FPR and FNR and highest AUC.

Table 2.2. Classification performance on Diabetes Data

Method	FNR	FPR	AUC
SVM	0.4379 (0.0047)	0.1266 (0.0033)	0.7177 (0.0021)
ADABOOST	0.4206 (0.0065)	0.1941 (0.0038)	0.6927 (0.0027)
NB	0.4070 (0.0054)	0.1709 (0.0038)	0.7110 (0.0026)
US	0.2958 (0.0061)	0.2365 (0.0050)	0.7339 (0.0022)
OS	0.3065 (0.0059)	0.2296 (0.0043)	0.7319 (0.0023)
ROSE	0.3075 (0.0076)	0.2411 (0.0069)	0.7257 (0.0029)
PRSVM	0.2857 (0.0068)	0.2895 (0.0053)	0.7840 (0.0035)

2.6.3 Application II: Wilt data

Wilt Data Set [52] is a high-resolution remote Sensing data. It consists of image segments generated by segmenting pansharpened images and contain spectral information from the Quickbird multispectral image bands and texture information from the panchromatic image band and was used to detect diseased trees. There are five features: GLCM-Pan, Mean-G, Mean-R, Mean-NIR and SD-Pan, meaning GLCM

mean texture (Pan band), Mean green value, Mean red value, Mean NIR value and Standard deviation (Pan band), respectively. The response variable has two states, "diseased trees" or "other land cover".

On the UCI Machine Learning Repository webpage for Wilt Data(`https://archive.ics.uci.edu/ml/datasets/wilt`), we downloaded a training set of 4339 samples and a test set of 500 samples. The training set contains 74 samples labeled as "diseased trees" class and 4265 labeled as "other land cover" class. The imbalance ratio is high and exceeding 1:58.

In our study, we refer to the "diseased trees" as the positive class and "other land cover" as negative class. The mean performance metrics of seven classifiers are reported in Table 2.3. SVM and Naive Bayes perform very poorly to predict the positive class due to the high imbalance ratio. Surprisingly AdaBoost works fine and even outperforms ROSE. All three resampling methods and PRSVM are able to overcome the highly imbalance problem and provide reasonable prediction for both classes while PRSVM has the largest AUC.

Table 2.3. Classification performance on Wilt Data

Method	FNR	FPR	AUC
SVM	0.9733	0.0032	0.5118
ADABOOST	0.3525	0.0274	0.8100
NB	0.8556	0.0319	0.5562
US	0.0870	0.2987	0.8072
OS	0.1759	0.2225	0.8008
ROSE	0.1932	0.5165	0.6452
PRSVM	0.1337	0.2300	0.8842

CHAPTER 3

PAIRWISE LEARNING FOR AUTISM SPECTRUM DISORDER IMBALANCED CLASSIFICATION

3.1 Overview

In this chapter, we perform a classification task on data from the Autism Brain Imaging Data Exchange (ABIDE) repository [12]. In real-world case analysis, the number of autism spectrum disorder (ASD) patients is much smaller than that of typically developed individuals, so we apply pairwise robust support vector machine (PRSVM) algorithm developed in previous chapter to classify autism spectrum disorder (ASD) patients. In the experiments of this project, the correlation matrix of functional magnetic resonance imaging (fMRI) data was used as a feature for classification. We compared the classification performance of PRSVM and other machine learning methods under various conditions, including different sample ratios, data dimensions and sample sizes. The experimental results show that PRSVM can accurately detect individuals with autism when the data is imbalanced and it can yield results that are superior to or comparable with other traditional classification methods in various scenarios. Furthermore, when compared to machine learning algorithms that perform good only on small datasets, our algorithm performs better on large datasets.

3.2 Introduction

In a broad sense, fMRI is a neuroimaging technique, which includes diffusion-weighted imaging (DWI), dynamic susceptibility contrast (DSC), dynamic contrast

enhanced (DCE), magnetic resonance spectroscopy (MRS), chemical shift–selective saturation (CHESS), blood oxygenation level dependent (BOLD), etc. The narrow definition of fMRI mainly refers to functional magnetic resonance imaging based on BOLD. Exploring the way the human brain works has always been the focus of scholars' research. Before the appearance of fMRI technology, electroencephalography, magnetoencephalography and nuclear medicine technology have been used to detect human brain activity.

In 1980, Roy and Sherrington [83] found that regional cerebral-blood flow could reflect the viability of neurons in corresponding area. Ogawa et al. [74] proposed BOLD in 1990, based on the BOLD effect, fMRI can be realized. In 1991, fMRI showing brain structure and function was demonstrated for the first time [4]. The rationale for fMRI brain imaging is that the increasing of local neuronal activity often leads to increased oxygen demand. The oxygen in oxyhemoglobin will produce a paramagnetic molecule called deoxyhemoglobin. During examination, accumulated deoxyhemoglobin can act as a local contrast agent to enhance local signal intensity. Thus, natural contrast agents can target task-relevant brain regions and visualize it with fMRI.

fMRI has a great impact on cognitive neuroscience. Since its discovery in 1990, fMRI has become one of the most commonly used techniques. Especially in the field of treating mental illness, fMRI has become a routine diagnostic tool. So far, fMRI has been used to discover the abnormal function of brain regions related to various mental diseases [88, 121, 125]. For classification studies on fMRI data, support vector machines (SVMs) and kernel SVMs were widely applied [11, 98].

In reality, the number of patients is much smaller than the number of normal people. For imbalanced fMRI datasets, traditional classification methods always tend

to overfocus on the majority class. So finding a good classifier for imbalanced fMRI datasets has become an important research topic for scholars. In recent years, a variety of machine learning algorithms have been used to deal with imbalanced fMRI data classification [91, 109].

Autism was discovered and named by Kanner [53] in 1943, and it is recognized as a special type of developmental disorder by the World Health Organization and the American Psychiatric Association. The current consensus is that deficits in social and verbal communication skills and repetitive stereotyped behaviors manifested before the age of three are the defining characteristics of children with autism.

ABIDE repository is a data-sharing initiative designed to advance research on autism spectrum disorder (ASD). The project collects and shares neuroimaging data from multiple authoritative institutions, and these datasets contain multiple modalities of data including structural MRI, functional MRI, magnetic resonance spectroscopy, and magnetic resonance diffusion tensor imaging. This initiative currently has two large-scale collections: ABIDE I and ABIDE II, of which ABIDE collected 1112 datasets, of which 539 were from autistic patients, 573 were from typical controls, and ABIDE II collected 1114 datasets, of which 521 from autistic patients, 593 from typical controls. The datasets we will use in this chapter come from ABIDE I.

With the open sharing of ABIDE data, many analyses of ABIDE have emerged. Various machine learning algorithms such as empirical bayes, logistic regression and SVM [5, 9, 118] are used to classify ASD patients. The experimental results of many articles show that when all samples are used for classification, machine learning algorithms usually only achieve an accuracy rate of 60% to 70% [118]. The machine learning algorithm can only achieve reliable accuracy when the total sample size is less than 100 [2]. With the rise and development of neural networks, more deep learning

algorithms are used to improve the accuracy of ABIDE data classification [44,90,119].

3.3 Methodology

In this project, the correlation coefficients between brain voxels of functional magnetic resonance imaging (fMRI) data was used as features. Principal component analysis will be used to reduced the number of features. Then PRSVM will be used to build a classification model.

3.3.1 Principal component analysis

We know that if there is a strong linear correlation between certain dimensions in the data, the information provided by the sample on these two dimensions will be repeated to a certain extent. So we hope that the dimensions of the input are uncorrelated. In addition, the dimension of the correlation matrix for brain voxels is too large. In order to reduce the number of features, we choose to use principal component analysis (PCA) to reduce the dimensionality of the data.

Given a set of n observations $X = \{x_1, x_2, \ldots, x_n\}$ with $x_i \in \mathbb{R}^p$, the original data can be regarded as a matrix with n rows and p columns. Assuming that the mean of each dimension of the original data is 0, we let this matrix multiply an $p \times p$ orthogonal transformation matrix W, where W consists of column vectors $\{W_1, W_2, \ldots, W_p\}$. Then the original data is transformed into a new coordinate system. To control the magnitude of the values of the transformed data, each column vector has $\|W_i\| = 1$. The matrix after dimensionality reduction is $T = XW$, each column vector in T is $\{t_1, t_2, \ldots, t_p\}$. In order to compute the transformation matrix W, we need to

compute the eigenvalues and eigenvectors of the covariance matrix

$$C = \frac{1}{p} X X^{\top}.$$

The eigenvectors could be combined into a change matrix W from left to right in the order of eigenvalues from large to small. Here, we can keep the eigenvectors with big eigenvalues to reduce the dimensionality. If W' is used to represent the change matrix after discarding the eigenvectors with smaller eigenvalues, where W' consists of column vectors $\{W'_1, W'_2, \ldots, W'_k\}$ and $k < p$.

$$T' = XW'$$

is the data after dimensionality reduction to k dimension.

3.4 PRSVM for autism brain imaging data

In Chapter 2 we have proposed pairwise robust support vector machine (PRSVM) algorithm to handle the task of imbalanced data classification by adapting the RSVC loss [28]

$$L_{\mathrm{RSVC}}(yf(x)) = \sigma^2 \left(1 - \exp\left(\frac{(1 - yf(x))_+^2}{\sigma^2}\right)\right)$$

to a pairwise learning framework.

We assume that the label of autistic patients is 1 and the label of typical controls is -1. In the classification of autism spectrum disorder, the binary classifier f should provide a result of $sign(f(x_{autistic\ patient})) = 1$ and $sign(f(x_{typical\ control})) = -1$. For each pair of observations (x_i, y_i) and (x_j, y_j), if x_i represents an autistic patient and x_j represents a typical control, a good real-valued f will produce the result such that $f(x_i) - f(x_j) > 0$; on the contrary, if x_i represents a typical control and x_j represents an autistic patient, it should be $f(x_i) - f(x_j) < 0$; when both x_i and x_j represent an

autistic patient or a typical control simultaneously, we have no expectations. We let $y_{ij} = (y_i - y_j)/2$ and write up the notation

$$y_{ij} = \begin{cases} 1, & \text{if } y_i = 1 \text{ and } y_j = -1; \\ -1, & \text{if } y_i = -1 \text{ and } y_j = 1, \end{cases}$$

if the algorithm has good performance, we can image that we will get $y_{ij}(f(x_i) - f(x_j)) > 0$.

When the sample sizes of autistic patients and typical controls are imbalanced, PRSVM combines the observations of the minority class and the majority class into pairs to enter the model. This method can effectively balance the influence of the two classes.

3.5 Experiments and results

In this research, we used $rois - ho$ and $rois - ez$ data set [75]. There are 1035 samples in total with 6216 features in $rois - ho$ [95] data set, 505 samples are labeled as autistic patients and 530 are labeled as typical controls. There are 1035 samples in total with 6786 features in $rois - ez$ data set [21], 505 samples are labeled as autistic patients and 530 are labeled as typical controls. In the experiments of this chapter, we set the label of autistic patients to 1 and typical controls label to -1.

SVM and AdaBoost are used to be compared with PRSVM. For PRSVM, we selected $\sigma = 1$ in all experiments. The packages and functions used in the experiments are the same as the experiments in Chapter 2. We randomly sample 1/2 of positive cases and negative cases to form a training set and the rest cases were test set, the number of repetitions for each experiment was 20.

3.5.1 Experiment I: data with varying dimensions

In the first experiment, we randomly sampled 100 and 500 samples from the autistic patients and typical controls respectively, leading the imbalance ratio to 1:5. PCA method was used to reduce the dimension by 10 and 50 principal components. The results in Table 3.1 and Table 3.2 show that when the imbalance ratio reaches 1:5, the average AUC of SVM, AdaBoost and Naive Bayes are close to 0.5, which means these models are ineffective in this case. PRSVM has the best performance on 10-dimensional and 50-dimensional datasets.

Table 3.1. Classification performance on $rois - ho$ data set

Input dimension	Method	FNR	FPR	AUC
10	PRSVM	0.4440 (0.0125)	0.4205 (0.0105)	0.6037 (0.0048)
	SVM	0.9120 (0.0075)	0.0455 (0.0048)	0.5213 (0.0020)
	AdaBoost	0.7060 (0.0063)	0.2095 (0.0049)	0.5422 (0.0035)
	NB	0.7440 (0.0115)	0.1640 (0.0095)	0.5460 (0.0030)
50	PRSVM	0.5610 (0.0093)	0.2890 (0.0085)	0.6259 (0.0050)
	SVM	0.6780 (0.0076)	0.1805 (0.0058)	0.5708 (0.0035)
	AdaBoost	0.7590 (0.0112)	0.1405 (0.0061)	0.5503 (0.0038)
	NB	0.7010 (0.0199)	0.1775 (0.0141)	0.5608 (0.0039)

Table 3.2. Classification performance on $rois - ez$ data set

Input dimension	Method	FNR	FPR	AUC
10	PRSVM	0.4070 (0.0097)	0.4920 (0.0095)	0.5831 (0.0045)
	SVM	0.9520 (0.0051)	0.0295 (0.0031)	0.5093 (0.0014)
	AdaBoost	0.7000 (0.0070)	0.2315 (0.0042)	0.5343 (0.0036)
	NB	0.7490 (0.0139)	0.1715 (0.0077)	0.5398 (0.0040)
50	PRSVM	0.5640 (0.0087)	0.2800 (0.0062)	0.6131 (0.0050)
	SVM	0.6560 (0.0082)	0.1895 (0.0047)	0.5728 (0.0038)
	AdaBoost	0.7780 (0.0089)	0.1465 (0.0063)	0.5378 (0.0032)
	NB	0.7100 (0.0197)	0.1890 (0.0144)	0.5505 (0.0037)

3.5.2 Experiment II: data with varying imbalance ratios

In real-world applications, it is difficult to collect a large number of patients data for a medical institution. Here we would like to see if our model could perform better if we just add samples from typical controls. We randomly sampled 100 from the autistic patients, and sampled 100, 250 and 500 from typical controls. We keep 10 principal components in this experiment. The results are shown in Table 3.3 and 3.4.

The results of experiment II show that when we investigated the impact of imbalance ratio by varying it from 1:1 to 1:2.5 and then to 1:5, the FPR and FNR of PRSVM do not change rapidly while other methods had poor performance. Ahe AUC of PRSVM is slightly increased indicates the PRSVM does provide slightly better models by solely increasing the samples from health controls.

Table 3.3. Classification performance on *rois-ho* data with varying imbalance ratios

Imbalance Ratio	Method	FNR	FPR	AUC
1:1	PRSVM	0.4110 (0.0161)	0.4390 (0.0192)	0.5969 (0.0066)
	SVM	0.3930 (0.0095)	0.4420 (0.0088)	0.5825 (0.0039)
	AdaBoost	0.4700 (0.0087)	0.4320 (0.0085)	0.5510 (0.0043)
	NB	0.4900 (0.0199)	0.4110 (0.0195)	0.5505 (0.0052)
1:2.5	PRSVM	0.4660 (0.0141)	0.3936 (0.0136)	0.5892 (0.0035)
	SVM	0.9830 (0.0026)	0.0076 (0.0012)	0.5047 (0.0012)
	AdaBoost	0.7770 (0.0061)	0.1592 (0.0036)	0.5319 (0.0027)
	NB	0.8390 (0.0062)	0.1040 (0.0047)	0.5285 (0.0023)
1:5	PRSVM	0.5080 (0.0098)	0.3470 (0.0075)	0.6058 (0.0027)
	SVM	1.0000 (0.0000)	0.0000 (0.0000)	0.5000 (0.0000)
	AdaBoost	0.9230 (0.0020)	0.0480 (0.0011)	0.5145 (0.0009)
	NB	0.9420 (0.0026)	0.0200 (0.0008)	0.5190 (0.0012)

Table 3.4. Classification performance on $rois-ez$ data with varying imbalance ratios

Imbalance Ratio	Method	FNR	FPR	AUC
1:1	PRSVM	0.4740 (0.0156)	0.4540 (0.0128)	0.5596 (0.0059)
	SVM	0.4040 (0.0104)	0.4870 (0.0102)	0.5545 (0.0036)
	AdaBoost	0.4810 (0.0095)	0.4940 (0.0099)	0.5195 (0.0042)
	NB	0.4420 (0.0162)	0.4970 (0.0174)	0.5305 (0.0035)
1:2.5	PRSVM	0.5390 (0.0127)	0.3660 (0.0090)	0.5688 (0.0042)
	SVM	0.9850 (0.0031)	0.0072 (0.0016)	0.5039 (0.0008)
	AdaBoost	0.7770 (0.0064)	0.1672 (0.0039)	0.5279 (0.0032)
	NB	0.8830 (0.0057)	0.0976 (0.0047)	0.5097 (0.0025)
1:5	PRSVM	0.4700 (0.0093)	0.4078 (0.0089)	0.6061 (0.0027)
	SVM	1.0000 (0.0000)	0.0000 (0.0000)	0.5000 (0.0000)
	AdaBoost	0.9470 (0.0023)	0.0446 (0.0012)	0.5042 (0.0009)
	NB	0.9720 (0.0021)	0.0148 (0.0008)	0.5066 (0.0008)

3.5.3 Experiment III: data with varying sample size

We further investigate if we can train a better model by increasing the number of samples in the training set while keeping the imbalance ratio constant. This is a seemly self-evident phenomenon in machine learning but failed to be seen in ASD study. In the third experiment, the sample size is increased proportionally but the imbalance ratio is fixed as 1:2. We randomly sampled 50, 150 and 250 data from positive class, 100, 300 and 500 from negative class, respectively. The results are shown in the Table 3.5 and Table 3.6.

We see that the performance of SVM, AdaBoost and Naive Bayes method degrades significantly and fails to detect patients. It verifies to a certain extent that machine learning algorithms do not perform well when the sample size is large, which coincides the findings in the literature. But for PRSVM, AUC becomes larger with the increase of sample size, showing that PRSVM is able to overcome this problem.

Table 3.5. Classification performance on $rois - ho$ data with varying sample sizes

Input data size	Method	FNR	FPR	AUC
25:50	PRSVM	0.4920 (0.0215)	0.4540 (0.0207)	0.5674 (0.0091)
	SVM	0.8580 (0.0129)	0.0740 (0.0055)	0.5340 (0.0048)
	AdaBoost	0.7020 (0.0144)	0.2310 (0.0070)	0.5335 (0.0074)
	NB	0.7600 (0.0173)	0.1780 (0.0163)	0.5310 (0.0044)
75:150	PRSVM	0.4730 (0.0116)	0.3687 (0.0090)	0.6195 (0.0023)
	SVM	0.9413 (0.0049)	0.0300 (0.0026)	0.5143 (0.0013)
	AdaBoost	0.7147 (0.0038)	0.2147 (0.0028)	0.5353 (0.0022)
	NB	0.7967 (0.0077)	0.1190 (0.0047)	0.5422 (0.0021)
125:250	PRSVM	0.3504 (0.0047)	0.4522 (0.0040)	0.6348 (0.0018)
	SVM	0.9820 (0.0020)	0.0096 (0.0011)	0.5042 (0.0005)
	AdaBoost	0.7112 (0.0028)	0.1984 (0.0022)	0.5452 (0.0012)
	NB	0.7690 (0.0026)	0.1136 (0.0020)	0.5452 (0.0009)

Table 3.6. Classification performance on $rois - ez$ data with varying sample sizes

Input data size	Method	FNR	FPR	AUC
25:50	PRSVM	0.5000 (0.0239)	0.4110 (0.0226)	0.5517 (0.0092)
	SVM	0.9000 (0.0106)	0.0560 (0.0057)	0.5220 (0.0038)
	AdaBoost	0.7020 (0.0180)	0.2490 (0.0110)	0.5245 (0.0067)
	NB	0.7960 (0.0174)	0.1640 (0.0099)	0.5200 (0.0078)
75:125	PRSVM	0.4467 (0.0096)	0.4316 (0.0085)	0.5963 (0.0021)
	SVM	0.9653 (0.0040)	0.0163 (0.0018)	0.5092 (0.0013)
	AdaBoost	0.6960 (0.0044)	0.2233 (0.0025)	0.5403 (0.0027)
	NB	0.7540 (0.0070)	0.1667 (0.0053)	0.5397 (0.0018)
125:250	PRSVM	0.4392 (0.0067)	0.4040 (0.0069)	0.6093 (0.0016)
	SVM	0.9888 (0.0018)	0.0088 (0.0014)	0.5012 (0.0002)
	AdaBoost	0.7152 (0.0022)	0.2188 (0.0017)	0.5330 (0.0010)
	NB	0.8140 (0.0039)	0.1194 (0.0026)	0.5333 (0.0009)

CHAPTER 4

KERNEL PAIRWISE ROBUST SUPPORT VECTOR MACHINE

4.1 Overview

Most of the data we encounter in real-world applications are high-dimensional and linearly inseparable. For linearly inseparable data, it is difficult to use a linear classifier to accurately classify it, so the kernel support vector machine comes into being. Due to the increasing need for classification of high-dimensional imbalanced datasets, many recent studies have focused on imbalanced classification using kernel support vector machines [16, 54, 108].

In Chapter 2, we have proposed a pairwise robust support vector machine for the linear imbalanced binary classification problem. In this chapter, we use kernel method to handle the nonlinear classification. The results of simulation and practical application show that our algorithm is highly effective.

4.2 Introduction

Nonlinear classification problems in the input space can be transformed into linear classification problems in a certain dimensional feature space through nonlinear transformation, and a linear support vector machine can be learned in a high-dimensional feature space. In 1995, Boser et al. [6] extended SVM to nonlinear problems, and realized nonlinear classification by using kernel functions to project data into high-dimensional spaces. The effectiveness of the algorithm was verified in handwritten digit recognition. The success of the kernel method on SVM has made it popular and developed rapidly. In 2001, Shawe-Taylor and Cristianini [93] gave a comprehen-

sive description of the kernel method in their book. Schölkopf et al. [86] proposed a principal component analysis algorithm based on kernel functions in 2002. In the following years, kernel methods have been widely used in regression problems [51,96], probability density estimation problems [72,105] and other fields.

Since in the dual problem of linear support vector machine learning, both the objective function and the classification decision function only involve the inner product, so there is no need to explicitly specify the nonlinear transformation, but replace the inner product with the kernel function. The kernel function represents the inner product between two instances after a nonlinear transformation. Specifically, $k(x,z)$ is a kernel function if there exists a mapping $\phi(x)$ from the input space to the feature space, and for any x,z in the input space, $k(x,z)=\phi(x)\phi(z)$. In the dual problem of linear support vector machine learning, the inner product is replaced by the kernel function $k(x,z)$, and the solution is the nonlinear support vector machine

$$f(x)=\text{sign}\left(\sum_{i=1}^{N}\alpha_i y_i k(x,x_i)+b\right).$$

Given the training data set $T=\{(x_1,y_1),(x_2,y_2),...,(x_N,y_N)\}$, where $x_i\in\mathbb{R}^n$, $y_i\in\{+1,-1\},i=1,2,...,N$, and selecting an appropriate kernel function $k(x,z)$ and penalty parameter $C>0$, the kernel support vector machine solves the convex quadratic programming problem

$$\min_{\alpha}\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j k(x_i,x_j)-\sum_{i=1}^{N}\alpha_i$$

$$s.t.\sum_{i=1}^{N}\alpha_i y_i=0$$

$$0\le\alpha_i\le C,i=1,2,...,N.$$

The goal is to get the optimal solution $\alpha^*=(\alpha_1^*,\alpha_2^*,...,\alpha_N^*)^T$. After getting the optimal solution α^*, choosing a component α_j^* of α^* that satisfies $0<\alpha_j^*\le C$,

compute

$$b^* = y_j - \sum_{i=1}^{N} \alpha_i^* y_i k(x_i, x_j).$$

So the classifier decision function is

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i^* y_i k(x, x_i) + b^*\right).$$

The Gaussian kernel function

$$k(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$

is a commonly used kernel function. The corresponding SVM is a Gaussian radial basis function classifier, in which case the classification decision function is

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i^* y_i \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right) + b^*\right).$$

4.3 Kernel PRSVM

In real tasks, there may not be a hyperplane that can correctly divide the two types of samples in the original sample space. We may consider to construct nonlinear classifiers using the kernel trick. Kernel support vector machine is a typical example of this idea. The purpose of this chapter is to extend this idea to PRSVM and propose kernel PRSVM to build nonlinear classifiers for imbalanced data. We will use Gauss kernel $k(x, z) = e^{\frac{-\|x-z\|^2}{2\sigma^2}}$.

Assume that the data are given as $\{x_1, x_2, ..., x_n\}$,

$$f(x) = \sum_{i=1}^{n} c_i k(x, x_i).$$

We still use pairwise RSVC loss

$$L(f, (x_i, y_i), (x_j, y_j)) = \sigma^2 \left(1 - \exp\left(\frac{(1 - y_{ij}(f(x_i) - f(x_j)))_+^2}{\sigma^2}\right)\right).$$

We define norm $\|f\|_k$ in function space $\mathcal{H}_k$, a reproducing kernel Hilbert space associated to the kernel function k, by

$$\|f\|_k^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i, x_j).$$

Then the kernel PRSVM algorithm could be written as

$$\min_{f\in\mathcal{H}_k} \frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j\neq y_i} L(f,(x_i,y_i),(x_j,y_j)) + \lambda\|f\|_k^2,$$

where $\lambda\|f\|_k^2$ as regularization term to control the capacity when the function space is too huge. The associated optimization problem is

$$\begin{aligned}\min_{(c_1,\ldots,c_n)\in\mathbb{R}^n} \quad & \frac{1}{n}\sum_{i=1}^{n}\sum_{j:y_j\neq y_i} \sigma^2\left(1-\exp\left(\frac{(1-y_{ij}\sum_{s=1}^{n} c_s(k(x_i,x_s)-k(x_j,x_s)))_+^2}{\sigma^2}\right)\right)\\ & +\lambda\sum_{s,t=1}^{n} c_s k(x_s,x_t)c_t,\end{aligned}$$

to solve the unknown coefficients $c_1,\ldots,c_n$. It can be implemented by the gradient descent algorithm.

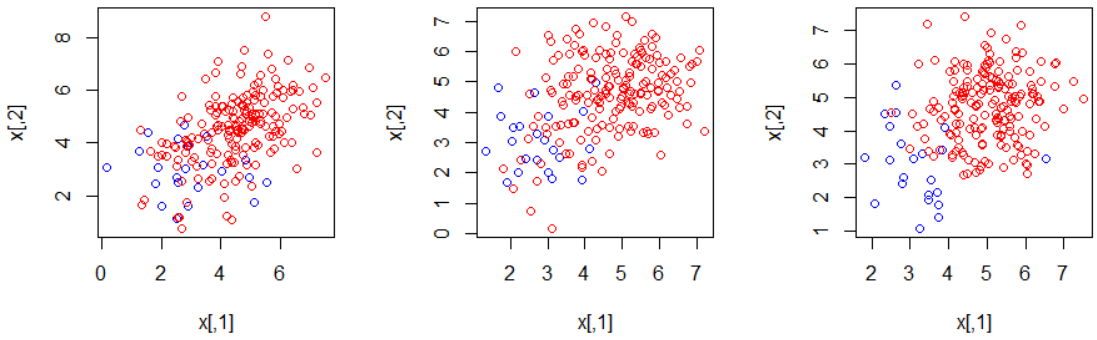
4.4 Simulations and applications

In this section, simulation studies and real-world applications are used to illustrate the effectiveness of kernel PRSVM (K-PRSVM) to handle imbalanced data. We will compare it with Kernel-SVM (K-SVM) method in simulation studies. In real-world applications we will compare it with both linear SVM and K-SVM.

Here we still selected $\sigma = 1$ in all experiments. For SVM and Kernel-SVM we used the `train` function in `caret` package. For fair comparison, method `svmlinear` is used to produce linear classifiers in SVM while `svmRadial` is used in Kernel-SVM. Other parameters are kept default. FPR, FNR and AUC are again used to evaluate the performance of the model.

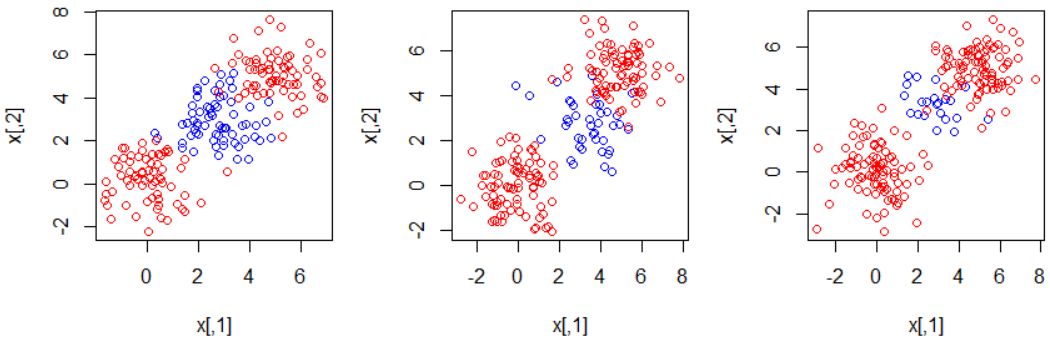
4.4.1 Simulation studies

In simulation studies, we assume the data come from $\mathbb{R}^2$ and generate two different datasets, a linearly separable one and a linearly inseparable one, for the experiments. The linearly separable dataset is the same as in 1.3. The positive class, $x_i = (x_{i,1}, x_{i,2})$ has both features $x_{i,1}$ and $x_{i,2}$ sampled from normal distribution with mean 3 and variance 1. For the negative class, both features are sampled from normal distribution with mean 5 and variance 1. For linearly inseparable data, on the basis of a linearly separable dataset, half of samples from the negative classes are shift down-left by subtracting 5 from each feature. In the simulation studies of this chapter, the positive class is still the minority class. To create data sets with imbalanced ratio $1 : k$, with training sample size n, we will generate $\frac{n}{k+1}$ samples for the positive class and $\frac{kn}{k+1}$ samples for the negative class. The training sets are illustrated in Figure 4.1 and Figure 4.2.



(a) $k = 2$ (b) $k = 4$ (c) $k = 8$

Figure 4.1. Linearly separable data with varying imbalance ratios



(a) $k = 2$ (b) $k = 4$ (c) $k = 8$

Figure 4.2. Linearly inseparable data with varying imbalance ratios

To compare the performance of seven methods, we fix the number of training samples $n = 200$ and investigate the impact of imbalance ratio by varying it from 1:2 to 1:4 and then to 1:8. Each experiment is repeated 50 times. The performance of each classifier is evaluated on test sets with 1000 samples from the positive class and $1000k$ samples from the negative class. The average FPR, FNR and AUC for the two classifiers are reported in Table 4.1 and Table 4.2. The results indicate that the advantages of K-PRSVM on linearly separable data sets are obvious, especially when the imbalance ratio reaches 1:8. The classification ability of Kernel-SVM for minority class is significantly reduced, but K-PRSVM can still get similar and relatively low FPR and FNR. K-PRSVM still performs better than Kernel-SVM on linearly inseparable data sets. As the imbalance ratio continues to increase, the results provided by K-PRSVM will not change rapidly. In all scenarios K-PRSVM achieves the larger and more stable AUC.

Table 4.1. Classification performance of K-PRSVM and Kernel-SVM on simulated linearly separable data with $n = 200$ and varying imbalance ratios

k	Method	FNR	FPR	AUC
1:2	K-PRSVM	0.0793 (0.0004)	0.0844 (0.0004)	0.9767 (<0.0001)
	K-SVM	0.1455 (0.0006)	0.0444 (0.0002)	0.9552 (0.0002)
1:4	K-PRSVM	0.0815 (0.0004)	0.0854 (0.0003)	0.9762 (<0.0001)
	K-SVM	0.2255 (0.0007)	0.0261 (0.0001)	0.9349 (0.0002)
1:8	K-PRSVM	0.0871 (0.0004)	0.0841 (0.0004)	0.9759 (<0.0001)
	K-SVM	0.3717 (0.0009)	0.0120 (<0.0001)	0.9088 (0.0002)

Table 4.2. Classification performance of K-PRSVM and Kernel-SVM on simulated linearly inseparable data with $n = 200$ and varying imbalance ratios

k	Method	FNR	FPR	AUC
1:2	K-PRSVM	0.0813 (0.0003)	0.1201 (0.0004)	0.9631 (<0.0001)
	K-SVM	0.1186 (0.0005)	0.0727 (0.0004)	0.9623 (0.0001)
1:4	K-PRSVM	0.0872 (0.0005)	0.1187 (0.0004)	0.9623 (0.0001)
	K-SVM	0.2057 (0.0006)	0.0395 (0.0002)	0.9546 (0.0001)
1:8	K-PRSVM	0.1040 (0.0004)	0.1005 (0.0003)	0.9636 (<0.0001)
	K-SVM	0.3241 (0.0007)	0.0207 (<0.0001)	0.9366 (0.0002)

4.4.2 Application on MINIST dataset

The MNIST data set is a classic benchmark data set in the field of machine learning. It consists of 60,000 training samples and 10,000 test samples. Each sample is a 28*28 pixel grayscale handwritten digital picture. The value of each pixel is between 0 and 255. Each picture represents a number from 0 to 9. By vectorizing the 28*28 matrix of each picture, a 784-dimensional vector can be obtained.

In this application, we merge the training and test sets of the original dataset to obtain a dataset of 70,000 samples and extract samples of the digits 3, 5, and 8.

Among them, there are 7141 pictures of the number 3, 6313 pictures of the number 5, and 6825 pictures of the number 8. We conducted three binary classification experiments: the pictures of the digit 3 as the positive class while 5 as the negative class, the pictures of digit 3 as the positive class while 8 as negative class, and the pictures of digit 5 as positive class while 8 as negative class. In each experiment, we set the imbalance ratio to $k = 4$. We randomly select 200 pictures and $200k$ pictures from the positive class and negative class respectively. 1/3 of the selected pictures are used as the training set, and the remaining pictures are used as the test set.

We reduced the dimensionality of the data by calculating the variance Var_{col}, $col = 1, 2, ..., 784$, of the training set in each dimension and eliminating features with variances less than $max(Var_{col})/100$. The average FPR, FNR and AUC for the K-PRSVM, SVM and Kernel-SVM are reported in Table 4.3.

In this application, we can see that although the Kernel-SVM method has a high AUC, it sacrifices the classification accuracy of the minority class. Its ability to detect minority samples is not even as good as SVM, and the K-PRSVM algorithm performs well on this high-dimensional dataset.

Table 4.3. Classification performance on MNIST data

digits	Method	FNR	FPR	AUC
3&5	K-PRSVM	0.1496 (0.0017)	0.0280 (0.0006)	0.9790 (0.0003)
	SVM	0.1899 (0.0014)	0.0288 (0.0005)	0.8907 (0.0007)
	K-SVM	0.4536 (0.0043)	0.0008 (<0.0001)	0.9837 (0.0003)
3&8	K-PRSVM	0.1451 (0.0021)	0.0320 (0.0007)	0.9791 (0.0003)
	SVM	0.1646 (0.00016)	0.0224 (0.0004)	0.9065 (0.0008)
	K-SVM	0.3704 (0.0032)	0.0039 (0.0002)	0.9818 (0.0003)
5&8	K-PRSVM	0.1441 (0.0029)	0.0316 (0.0005)	0.9849 (0.0002)
	SVM	0.1729 (0.0015)	0.0328 (0.0005)	0.8972 (0.0007)
	K-SVM	0.5555 (0.0032)	0.0025 (0.0001)	0.9797 (0.0003)

CHAPTER 5

ROBUST REPRESENTATIONS IN DEEP LEARNING

5.1 Overview

Deep neural networks are playing increasing roles in machine learning and artificial intelligence to handle complicated data. The performance of deep neural networks depends highly on the network architecture and the loss function. While the most common choice for loss function is the squared loss for regression analysis it is known to be sensitive to outliers and adversarial samples. To improve the robustness, we introduce the use of the correntropy loss to the implementation of deep neural networks. We further split the neural network architecture into a feature extraction component and function evaluation component and design four two-stage algorithms to study which component is more impacted by the use of the robust loss. The applications in several real data sets indicates that the robust deep neural networks can efficiently generate robust representations of complicated data and the two-stage algorithms are consistently more powerful than their one-stage counterparts.

5.2 Introduction

The history of artificial neural networks dates back at least to the perceptron invented by Rosenblatt [81] in the 1950s. After more than half a century's development, artificial neural networks are playing increasing roles in modern machine learning and artificial intelligence applications. Although it has been proved that the feedforward neural network with a single hidden layer and the sigmoid activation function can approximate any arbitrarily complex continuous mapping with arbitrary

precision [13, 31, 46], more and more evidences show that deep neural networks could be more powerful [36, 61, 85]. In the past decade, along with the fast development of hardwares and computational powers, deep neural networks have been successfully applied to computer vision, speech and audio recognition, language processing, customer relationship management, and many other fields.

The performance of deep neural networks highly depends on the network architecture and the loss function. In the context of supervised learning, three main neural network architectures are popularly used. The fully connected feedforward neural networks, the convolutional neural networks, and the recurrent neural networks. While fully connected neural networks could be more widely applied to any structured dataset, convolutional neural networks have been shown powerful for image analysis and computer vision, and recurrent neural networks have been successfully used in time series data such as speech recognition and natural language processing. Regarding the loss functions, the least square loss and cross-entropy loss are commonly used for regression analysis and classification tasks, respectively.

Robustness concerns may arise in practice when the data is contaminated by outliers and adversarial samples. For instance, rare body poses in human pose estimation, unlikely facial point position in facial landmark detection, imprecise ground-truth annotations, and label misspecification, all may result abnormal samples in image processing; outliers may present in financial data due to heavy tailed distributions; and system shock could produce extreme and erratic values in signal processing. In these situations there are needs to develop robust deep learning approaches because least square loss are well known to be unrobust and sensitive to outliers. Some efforts have been done in the literature, e.g. [8, 37, 68]. While there are multiple ways to promote algorithm robustness, the most common approach is to adopt a robust loss

to train the neural networks. Typical examples of robust losses include the Huber's loss, the Tukey's biweight loss, the truncated least square loss, the Cauchy loss, and the correntropy loss.

In this chapter, we propose to build robust deep neural networks using the correntropy loss. We will not only verify its effectiveness, but also thoroughly explore where robustness comes from. To be precise, recall that a deep neural network is usually regarded as the combination of two components, the feature extraction component and the function evaluation component. We are particularly interested in the impact of the robust loss on these two components and will evaluate if both components are impacted or one is more impacted than the other. In order to make fair comparisons, we design two-stage algorithms and conduct a comparative study on real world applications. The results indicate two interesting findings: (1) While the robust loss may impact both components, it seems the feature extraction part is more impacted. In other words, robust deep neural networks incline to produce more robust feature representations. (2) The two-stage implementations of deep neural networks are always more powerful than the single-stage approaches, regardless of the loss functions used.

5.3 Robust deep neural networks

In this subsection, we introduce two robust deep neural networks, the robust deep feedforward neural network and the robust long short-term memory neural network.

5.3.1 The correntropy loss

Outliers are abnormal or extreme values in the data that significantly deviate from the rest of the observations. The appearance of a small amount of outliers may reduce

the ability of statistical inference and hurt the predictive performance of machine learning models. While outlier detection and removal are used sometimes [30] [120], a more common approach for supervised learning is to adopt a robust loss function.

Given a data set $(\boldsymbol{x}_i, y_i), i = 1, \ldots, n$ with $\mathbf{x}_i \in \mathbb{R}^d$ representing the vector of d explanatory variables and y_i the response, the well-known least square method aims to minimize the the mean square error

$$\min_f \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where $\hat{y}_i = f(\boldsymbol{x}_i)$ is the prediction of the response variable y_i by a hypothetical function f. The minimization process is conducted over a set of hypothesis functions which could be the set of linear functions in the traditional multiple linear regression or the set of nonlinear functions represented by a neural network architecture. A main advantage of the least square method is its optimality when the noise follows a Gaussian distribution while the main criticism is the lack of robustness when the Gaussianity is violated by outliers of heavy tailed distributions.

The use of correntropy loss for robustness has a long history. Its variant forms have been proposed as goodness-of-fit measures in the literature under different terminologies, such as the Welsch's loss [18], the inverted Gaussian loss [55], the exponential squared loss [111], the reflected normal loss [99], and the maximum correntropy criterion [66] [76]. In this chapter we adopt the form proposed in [25]:

$$\mathcal{L}(y_i, \hat{y}_i) = \sigma^2 \left(1 - \exp\left(-\frac{(y_i - \hat{y}_i)^2}{\sigma^2}\right)\right),$$

where $\sigma > 0$ is a tunable parameter that trades off the robustness and fitting errors. A robust regression approach minimizes the mean correntropy loss

$$\min_f \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, \hat{y}_i).$$

There exist not only numerous empirical evidences in the literature to show the ability of correntropy loss to promote robustness, the theoretical guarantees were also investigated in recent studies [24, 26, 27, 29].

5.3.2 Robust deep forward neural network

A fully connected Forward Neural Network (FNN) consists of three parts: the input layer, the hidden layers, and the out put layer. The input layer has d neurons, representing the d features of the input data. Mathematically, for an input vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$, the jth node of the input layer is given by $h_j^0 = x_j$. A FNN can have one or multiple hidden layers. It is called a shallow neural network if there is only one hidden layer and a deep neural network if there are two or more hidden layers. As we have already mentioned above, although a shallow neural network has the ability to well approximate arbitrarily complicated functions, there are both empirical and theoretical evidence that deep neural networks are more powerful in real applications. For hidden layers, the value of each neuron is computed from all neurons of the precedent layer by an affine linear mapping and an activation function: let $h_{l,j}$ denote the j-th node of the l-th layer and d_l be the number of neurons in the l-the layer. Then

$$h_{k,j} = a\left(\sum_{j=1}^{d_{l-1}} w_{l,j,k} h_{l-1,j} + b_{l,j}\right),$$

where $w_{l,j,k} \in \mathbb{R}$, $b_{l,j} \in \mathbb{R}$, and a is an activation function. The most popular choices for the activation function include the sigmoid function, the hyperbolic tangent function, and the rectified linear activation function (ReLU). The output layer of an FNN will produce predicted values for the response variable. For a regression analysis with a scalar response variable, the output layer contains one neuron by linear function of

the last hidden layer:

$$\hat{y} = \sum_{j=1}^{d_L} w_{L,j} h_{L,j} + b_L,$$

where L denotes the number of hidden layers. Figure 5.1 shows an example of FNN with two hidden layers and a single output. The number of output neurons can be more than one for vector valued regression analysis or classification problems.

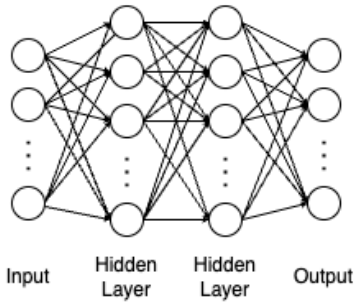


Figure 5.1. A deep forward neural network with two hidden layers

The training of the weight and bias parameters of FNN requires a loss function to measure the error when $\hat{y}_i$ is used to predict the true response value y_i for each observation $\mathbf{x}_i$. In this chapter, we implement the robust deep forward neural network (RFNN) by minimizing the mean correntropy loss.

5.3.3 Robust long short-term memory neural network

Long short-term memory neural networks (LSTM) was first introduced by Hochreiter and Schmidhuber [45]. In 1999, Felix et al. [34] introduced a forget gate mechanism based on Hochreiter and Schmidhuber's work, which enables LSTM to reset its own state to avoid network crashes. Several variant models were proposed since then. LSTM is a special kind of recurrent neural network and has been shown effective for time series analysis, speech recognition, language translation, and natural language

processing due to its ability to memorize long and short term information.

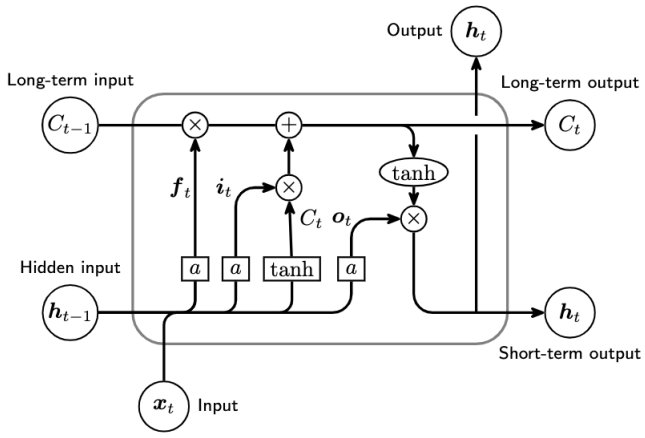


Figure 5.2. LSTM network structure

Figure 5.2 shows the core structure of LSTM. The long term information is stored in the cell states and pass through the entire chain system of LSTM from beginning to end. Three layers will used to decide what information will be removed and what information will be added. The forget gate layer generate $\boldsymbol{f}_t$ of a vector of values between 0 and 1 based on the current input $\boldsymbol{x}_t$ and previous moment output h_{t-1} via affine linear transforms and sigmoid activation function:

$$\boldsymbol{f}_t = a\left(W_f \boldsymbol{x}_t + U_f \boldsymbol{h}_{t-1} + b_f\right)$$

where $W_{\boldsymbol{f}}$ is a matrix of weight coefficients and b_f a sequence of biases. The values of $\boldsymbol{f}_t$ determine the percentage of information in C_{t-1} that are allowed to pass through, in other words, the information $(1 - \boldsymbol{f}_t) * C_{t-1}$ will be removed or "forgotten", where $*$ denotes element wise multiplication operator . A tanh layer will produce values representing candidate information :

$$\tilde{C}_t = \tanh\left(W_C \boldsymbol{x}_t + U_C \boldsymbol{h}_{t-1} + b_i\right),$$

and the input gate layer produced $\boldsymbol{i}_t$, again a vector of values between 0 and 1, by

$$\boldsymbol{i}_t = a\left(W_i \boldsymbol{x}_t + U_i \boldsymbol{h}_{t-1} + b_i\right)$$

to decide the percentage of candidate information $\tilde{C}_t$ to be added to the cell state. The cell state is then updated by

$$C_t = \boldsymbol{f}_t * C_{t-1} + \boldsymbol{i}_t * \tilde{C}_t.$$

After the cell state is updated, LSTM will use the output gate will first compute

$$\boldsymbol{o}_t = a(W_o \boldsymbol{x}_t + U_o \boldsymbol{h}_{t-1} + b_o)$$

and then use $\tanh(C_t)$ as weight coefficients to generate the output

$$\boldsymbol{h}_t = \boldsymbol{o}_t * \tanh(C_t),$$

which will be further used to produce the prediction of the response variable. In this chapter a linear function

$$\hat{y}_t = \boldsymbol{w}^\top \boldsymbol{h}_t + b$$

is used. Given a sequence of time series $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$ and corresponding response series $y_1, y_2, \ldots, y_T$, the robust LSTM will minimize the mean correntropy loss

$$\frac{1}{T} \sum_{i=1}^{T} \mathcal{L}(y_t, \hat{y}_t)$$

over the historical period to estimate the network parameters.

5.4 Two-stage algorithms

When the data are contaminated by outliers or are skewed and have heavy tails, robust algorithms are supposed to perform better. As we will see in Section 5.5 below,

our robust deep learning algorithms are indeed superior as expected when they are applied to real applications with robustness concerns.

The success of deep neural networks have been largely attributed to its ability to extract information from the complicated data. Therefore, it is commonly recognized that a deep neural network can be split into two parts: a feature extraction part and a function evaluation part, where the first part extract relevant features from the input data and second part use the features to build a decision function. As we are able to show the superiority of robust deep learning algorithms, we want to explore further and answer the questions that (1) "whether the robust deep learning algorithms promote robustness of feature extraction and lead to robust representation?" and (2) "which part of the network is more impacted by the use of robust loss?" To answer these questions, we propose a series of two stage algorithms.

For FNN and RFNN, we regard the part from the input to the last hidden layer as the feature extraction process and from the last hidden layer to output as the function evaluation part. We first run FNN and robust FNN to build two neural networks. Then we extract the features and run the linear regression with either least square (LS) approach or the robust regression (RR) with correntropy loss. This leads to four two-stage algorithms: FNN+LS, FNN+RR, RFNN+LS, and RFNN+RR, where the FNN+LS uses the features extracted from FNN and least square regression to predict the response variable, FNN+RR uses the features extracted from FNN and robust regression, RFNN+LS uses the features extracted from RFNN and least square regression, and RFNN+LS uses the features extracted from RFNN and robust regression. If RFNN+LS outperforms FNN+LS, we are able to conclude that the feature extraction process by RFNN is robust. Otherwise the correntropy loss does not robustify the feature extraction process. On the other hand, if RFNN+RR outperforms

RFNN+LS, the correntropy loss plays a role in the function evaluation process.

In LSTM we regards the values in output h_t are the features extracted from the input x_t and previous information. We can similarly design four two-stage algorithms to study the robust representation ability of RLSTM.

5.5 Applications

In this section, we apply our algorithms to real-world applications and illustrate their effectiveness.

5.5.1 Airfoil dataset

Airfoil Self-Noise Data [20] is a NASA data set which obtained from a series of aerodynamic and acoustic tests of two and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel. It is a multivariate data set with 5 attributes (Frequency, Angle of attack, Chord length, Free-stream velocity and Suction side displacement thickness) measuring scaled sound pressure level. The data set contains 1503 instances. Figure 5.3 shows the histogram of the response variable. It is clearly left skewed.

We randomly sampled 50% (Different split ratios do not significantly affect the final results) of the data as training set and remaining data as test set. We apply FNN, RFNN, and all four two-stage algorithms to build models and predict on the test set. The neural network contains two hidden layers with each hidden layer containing 64 hidden neurons. Tensorflow in Python is used to train the neural network with both the epoch and batch sizes selected as 50. The parameter σ is not sensitive and a value of $\sigma = 10$ is used. The experiments are repeated 50 times. The average mean

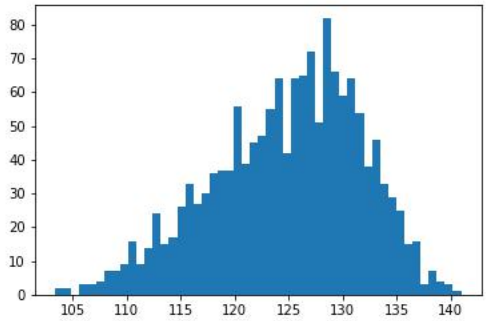


Figure 5.3. Histogram of response variable for airfoil data

absolute error (MAE) and the standard error (SE) of all six approaches are reported in Table 5.1.

Firstly, we see that RFNN outperforms FNN, indicating the use of correntropy loss improves the robustness of the neural network estimation. Next, RFNN+LS outperforms FNN+LS and RFNN+RR outperforms FNN+RR, that is, when the same regression approach is used, using features from RFNN is always better. This means that the features extracted by RFNN is more informative and therefore we can claim that the RFNN helps to extract features more robustly. Thirdly, FNN+RR outperforms FNN+LS and RFNN+RR also outperforms RFNN+LS, but the improvement is not significant. This means that once the features have been extracted, further use of robust loss in the regression step does not help much. Lastly, it is surprising to see that FNN+LS outperforms FNN and RFNN+RR outperforms RFNN, indicating that when the same loss function is used, the two-stage algorithms are consistently better than traing the neural network directly. Further more, if we change the loss function in the second regression stage, the two-stage algorithms are still better.

5.5.2 Boston housing dataset

Boston Housing Data Set [43] contains information collected by the U.S Census Service concerning housing in the area of Boston Massachusetts. It is a multivariate data set with 13 attributes measuring the median value of owner-occupied homes. It contains 506 instances. Figure 5.4 show the histogram of the home values. We can see outliers on the right end.

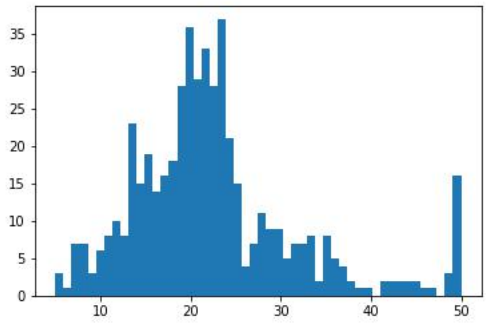


Figure 5.4. Histogram of home values in Boston housing data

This data has been build in the sklearn module in Python where the training set and test set have been automatically separated. We merged them together and then randomly sampled 50% as training set and put the remaining data into the test set. The hyperparameters and analysis process are the same as the experiment for Airfoil data. The results are shown in Table 5.1. The findings are very similar to the application in Airfoil data: RFNN is more robust than FNN. The use of correntropy seems play more roles in robust feature extraction while less roles in function evaluation. A follow-up regression stage helps further improve the performance.

5.5.3 Agroecosystem data

This data is collected by Dr. Song Cui at the MTSU Department of Agriculture. It contains carbon, water and energy fluxes of a cool-season dominated pasture ecosystem for 159 days from September 2, 2016 to May 3, 2017. For each day, there are 48 data points with each data point a summary of the relevant information in half an hour. If the data is complete, there should be 7632 data points in total. But due to power outage or system failure, a small number of data points are missing and the data actually contains 7265 data points. In this analysis, 12 driver variables that measure the radiation, humidity, temperature, wind, and some other features, were used to predict evapotranspiration flux. Outliers due to system shocks can be clearly seen from plot in Figure 5.5.

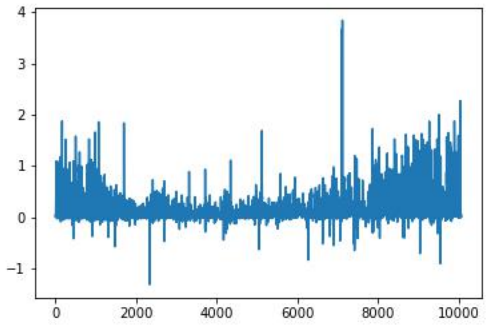


Figure 5.5. Plot of evapotranspiration flux

The deep neural network with two hidden layers and 64 hidden neurons for each layer is again used for the analysis. A 50%-50% split is again used for training and test data split. The RFNN used $\sigma = 50$. The experiments are repeated 50 times and the average MAEs are reported for all six approaches in Table 5.1. Similarly, we find the RFNN is able to lead to robust representation of the data and two-stage

algorithms are more powerful.

5.5.4 CSI 300 data

Per Wikipedia (`https://en.wikipedia.org/wiki/CSI_300_Index`), the CSI 300 is a capitalization-weighted stock market index designed to replicate the performance of the top 300 stocks traded on the Shanghai Stock Exchange and the Shenzhen Stock Exchange. It is a gauge of Chinese stock market. In this experiment, the trading information (opening price, closing price, highest price, lowest price and volume) of CSI 300 from January 3, 2017 to December 29, 2018 (excluding weekends and holidays) used. Figure 5.6 show the closing prices. Stock prices are typical examples of time series involving sudden changes and abnormal values due to the reaction to government policies, economical indicators, and sentiments.

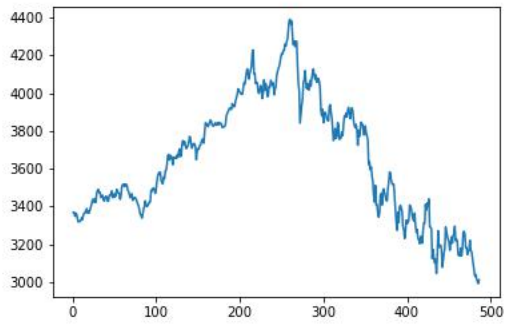


Figure 5.6. Histogram of closing prices in CSI 300

The analysis aims to predict the closing price based on the opening price, the previous day's opening price, closing price, highest price, lowest price and volume. As the price data can be viewed as time series, LSTM is appropriate. The results by six approaches are reported in Table 5.2. Although a different network architecture

Table 5.1. MAE on Airfoil and Boston housing data

Method	Airfoil	Boston Housing	Agroecosystem
FNN	0.2366 (0.0023)	0.2761 (0.0045)	0.1877 (0.0026)
FNN+LS	0.2235 (0.0016)	0.2719 (0.0040)	0.1720 (0.0007)
FNN+RR	0.2223 (0.0016)	0.2702 (0.0040)	0.1719 (0.0007)
RFNN	0.2279 (0.0018)	0.2706 (0.0035)	0.1779 (0.0014)
RFNN+LS	0.2173 (0.0014)	0.2681 (0.0035)	0.1714 (0.0009)
RFNN+RR	0.2161 (0.0014)	0.2669 (0.0035)	0.1713 (0.0008)

is used in this experiment, the findings are still consistent with previous applications. The only difference is that the use of robust regression in the second stage play more roles, as is evidenced the better performance of LSTM+RR and RLSTM+RR than that of LSTM+LS and RLSTM+LS, respectively.

Table 5.2. MAE on AIU and CSI300 data

Method	CSI300
LSTM	0.2221 (0.0007)
LSTM+LS	0.2133 (0.0007)
LSTM+RR	0.2016 (0.0006)
RLSTM	0.2197 (0.0008)
RLSTM+LS	0.2116 (0.0007)
RLSTM+RR	0.2012 (0.0007)

CHAPTER 6

ROBUST ADVERSARIAL ATTACKS ANALYSIS IN CONVOLUTIONAL NEURAL NETWORK

6.1 Introduction

When deploying an image classification model in real-world applications, we want the model not only to perform well but also to be resistant to intentionally malicious attacks. Malicious attacks will make the self-driving car off the route [59], the target detection error [116] and the face recognition system invalid [92]. In order to avoid such situations, we need to improve the robustness of the image classification model. An important method to achieve this goal is to use adversarial examples. Adversarial examples show that the model is not robust to input noise, which leads to two concepts: "against attack" and "against defense". "Adversarial attack" is to find a way to create more adversarial examples, so that the model will produce wrong results when predicting, and "adversarial defense" is to find a way to allow the model to correctly identify more adversarial examples. "Adversarial training" is a type of "adversarial defense". It constructs adversarial examples and adds them to the training set to enhance the robustness of the model, thereby improving the generalization ability of the model. When generating adversarial examples, the 0-1 loss is not easy to optimize with the gradient method, so we choose to use correntropy loss instead.

6.1.1 Convolutional neural networks

A convolutional neural network (CNN) is a feedforward neural network. It is one of the representative algorithms of deep learning. In recent years, it became a

confidential algorithm due to its excellent performance in tasks of image recognition systems. Research on CNNs began in the 1980s to 1990s. Time delay neural networks (TDNN) is the earliest CNNs [107], which was used in voice recognition. The speech signal preprocessed by fast Fourier transform was used as input. The hidden layers consist of two one-dimensional convolution kernels which are used to extract shift invariant features in the frequency domain. Before the emergence of TDNN, researchers made breakthrough in Back-Propagation (BP) algorithm [84]. Research about TDNN was developed within the framework of BP algorithm. Zhang [123] proposed the first two-dimensional CNN, shift-invariant artificial neural networks (SIANN). This neural network was used in medical image testing. LeCun et al. proposed a CNN applied to computer vision problems [62], which is the initial version of LeNet. They used stochastic gradient descent algorithm in their network. LeNet contains two convolutional layers and two fully connected layers. Its scale far exceeds TDNN and SIANN and its structure is very close to modern CNNs [64]. Based on LeNet, LeCun et al. [63] proposed a more complete CNN LeNet-5. This CNN got good performance in the recognition of handwritten digits. Due to the success of LeNet-5, the application of CNNs got more attention. Research based on CNNs such as face recognition [60], gesture recognition [33] has also been launched. With the improvement of deep learning theory and the update of numerical calculation equipment, CNNs have developed rapidly and are widely used in computer vision and natural language processing such as image classification and retrieval [115], target location and detection [124], face and skeleton recognition [40].

6.1.2 Adversarial attacks

Adversarial attacks originated from image attacks, then gradually improved the theory and expanded to video attacks and applications in natural language processing, reinforcement learning and other fields. Adversarial attacks are divided into the several types. White-box attacks, also known as open-box, means that in this case the attacker has full knowledge of the model and the training set. This case is relatively simple, but it does not match the actual situation. Black-box attacks means that the attacker has no knowledge of the model, little or no knowledge of the training set. This kind of attacks is more in line with the actual situation, and it is also the main research direction. A targeted attack refers to a multi-classification network that misjudges the input classification to a specified class. The non-target attack only needs to generate adversarial examples to deceive the neural network, which can be regarded as a special case of the above. In image attacks field, adversarial attacks usually add some subtle perturbations to the input examples, causing the model to give a false output with high confidence.

Now adversarial attacks are methods that specifically study how to improve the robustness of network models. But adversarial attacks are often used in classification problems and there is less discussion about its use in regression problems. We know that in linear regression, parameters are updated by minimizing the loss function, in an adversarial attack, the goal is to maximize the loss function. Small disturbances invisible to the naked eye will be amplified by linear mapping. So in this chapter, we use a non-target attack to check if adversarial attacks could also strongly affect the regression questions, and we plan to further look at the impact of adversarial attacks on neural networks of different architectures.

In 2013, Szegedy et al. [101] proposed the concept of adversarial examples. Some

properties of neural networks were introduced in the paper. The authors pointed out that the input-output mapping of deep neural network learning is discontinuous, and some imperceptible perturbations can cause the network to misclassify images. This disturbance is discovered by maximizing the prediction error of the network. Furthermore, the specific nature of these perturbations is not a random product of learning, and the same perturbation can cause different networks trained on different subsets of the data set to misclassify the same input. Memory Broyden-Fletcher-Goldfarb-Shanno (BFGS), the earliest adversarial attack algorithm, was also proposed in this paper. This algorithm converts the problem into a convex optimization by finding the smallest loss function addition that causes the neural network to misclassify. The mathematical formulation of the problem is as follows:

$$\min \|r\|_2$$

$$s.t. \quad f(x+r) = l$$

$$x + r \in [0,1]^m,$$

where $f(x)$ represents the classification function, r represents the changing step size, and the formula expresses the search for the smallest r that makes $f(x+r)$ mapped to the specified class l.

Afterwards, on this basis, various improvements were proposed. In 2014, Goodfellow et al. [38] proposed the fast gradient sign method (FGSM) to generate adversarial examples with a single gradient step. This method is used to perturb the input of the model before backpropagation. It is an early form of adversarial training. Based on FGSM, Madry et al. [67] proposed Projected Gradient descent (PGD), which could deal with all first order attacks.

6.2 Adversarial attacks in convolutional neural networks

In this part we try to find out how robust loss affect the performance of an adversarial attack.

6.2.1 Architecture of convolutional neural networks

Our CNNs take an image as input and classify the label. The architecture of the network is shown in Figure 6.1. It consists of two convolutional layers, followed by two fully connected layers. We use mean square error (MSE) and correntropy loss to train

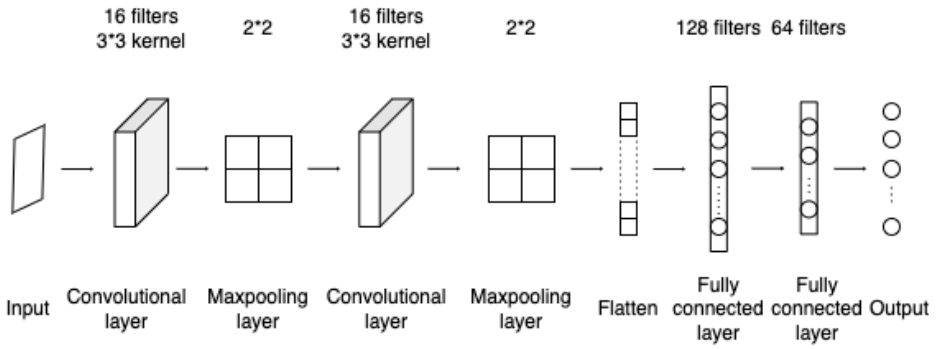


Figure 6.1. The structure of convolutional neural networks for experiments

the model respectively. Each experiment is repeated 20 times. We find no significant difference between the results of the two models. We then further investigate the performance of a CNN trained with robust loss functions when predicting adversarial examples. During the experimental process, we use different loss functions based on the FGSM method to generate the adversarial examples.

6.2.2 Fast gradient sign method

In this object, we utilize FGSM to generate the adversarial examples. The basic idea behind FGSM is to generate attack noise using gradients. For an original sample x, let the adversarial example be denoted as $x^{'}$, with $x^{'} = x + \eta$. In order for the adversarial example to remain unrecognized by the machine, the perturbation η should be very small. Hence, we set a limit on η, $\|\eta\|_{inf} < \epsilon$. Considering the linear transformation from the input layer to the adversarial examples:

$$w^T x^{'} = w^T x + w^T \eta$$

After a linear map, although the value of $\|\eta\|_{\infty}$ remains unchanged, the final result varies significantly due to factors such as increased dimensionality, transformation of matrix weights, and non-linear activation.

The way FGSM generates adversarial examples could be represented as follows.

$$x^{'} = x + \epsilon sign(\bigtriangledown_x J(\theta, x, y)),$$

where ϵ is a parameter that controls the level of perturbation or noise introduced, $J()$ represents the loss function, x and y denote the input image and true label, and θ represents the network parameters. In our experiments, we set different values of ϵ to generate the corresponding adversarial examples. Subsequently, we employ the models trained earlier to predict the labels of the adversarial examples.

6.3 Applications

In the application analysis, we employed mean square error (MSE), Categorical-Crossentropy (CC), and correntropy loss to train the model and generate adversarial

examples. Two real-world datasets were utilized in the experiments. We used TensorFlow in Python to train the neural networks, with the epoch selected as 50 and the batch size as 256. Each experiment was repeated 20 times for reliable results.

6.3.1 Application I: MINIST dataset

The MINIST data set [17] is used to evaluate the performance of the CNN trained with a robust loss. The classification accuracy for the adversarial examples is presented in Table 6.1.

The results indicate that as the value of ϵ increases, the classification accuracy of three models decreases rapidly. When $\epsilon = 1.5$, the accuracy for adversarial examples generated using the robust loss is lower compared to those generated using MSE and CC loss. However, when handling adversarial examples, the network trained with the robust loss demonstrates significantly better performance than the network trained with CC loss and slightly outperforms the network trained with Mean Square Error.

Table 6.1. Classification Accuracy for Adversarial Examples on MINIST dataset

Loss in CNN	Loss in attacks	$\epsilon = 0$	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 1.5$
MSE	MSE	0.9906 (0.0002)	0.9172 (0.0022)	0.6968 (0.0083)	0.4172 (0.0108)
	CC	0.9911 (0.0002)	0.9135 (0.0025)	0.7668 (0.0055)	0.5999 (0.0084)
	Robust	0.9913 (0.0002)	0.9156 (0.0030)	0.6916 (0.0112)	0.4091 (0.0146)
CC	MSE	0.9915 (0.0002)	0.9166 (0.0019)	0.6707 (0.0077)	0.3885 (0.0096)
	CC	0.9914 (0.0003)	0.9199 (0.0023)	0.7841 (0.0067)	0.6102 (0.0091)
	Robust	0.9918 (0.0003)	0.9179 (0.0021)	0.6785 (0.0074)	0.3856 (0.0093)
Robust	MSE	0.9912 (0.0002)	0.9183 (0.0019)	0.7059 (0.0067)	0.4392 (0.0083)
	CC	0.9911 (0.0002)	0.9210 (0.0020)	0.7877 (0.0048)	0.6309 (0.0086)
	Robust	0.9905 (0.0002)	0.9149 (0.0021)	0.6948 (0.0065)	0.4158 (0.0080)

6.3.2 Application II: CIFAR-10 dataset

The CIFAR-10 dataset [58] consists of a total of 60,000 32×32 color images, divided into 10 classes with 6,000 images in each class. There are 50,000 images for training, which are split into 5 training batches, each containing 10,000 images, and an additional 10,000 images for testing. The classification results are presented in Table 6.2.

For more complex images with three channels, it can be observed that the model trained with the robust loss performs similarly to the model trained with MSE in defending against attacks, but still outperforms the model trained with CC loss. Regarding adversarial examples, the conclusions obtained are consistent with the previous experiment.

Table 6.2. Classification Accuracy for Adversarial Examples on CIFAR-10 dataset

Loss in CNN	Loss in attacks	$\epsilon = 0$	$\epsilon = 0.001$	$\epsilon = 0.0025$	$\epsilon = 0.005$
MSE	MSE	0.6729 (0.0015)	0.6052 (0.0015)	0.5058 (0.0017)	0.3692 (0.0021)
	CC	0.6723 (0.0017)	0.6040 (0.0015)	0.5069 (0.0015)	0.3703 (0.0021)
	Robust	0.6734 (0.0014)	0.6042 (0.0013)	0.5056 (0.0015)	0.3672 (0.0018)
CC	MSE	0.6744 (0.0012)	0.5974 (0.0009)	0.4898 (0.0010)	0.3422 (0.0016)
	CC	0.6743 (0.0015)	0.6000 (0.0013)	0.4930 (0.0015)	0.3467 (0.0021)
	Robust	0.6730 (0.0014)	0.5957 (0.0014)	0.4875 (0.0018)	0.3409 (0.0021)
Robust	MSE	0.6724 (0.0016)	0.6045 (0.0014)	0.5065 (0.0015)	0.3685 (0.0019)
	CC	0.6722 (0.0012)	0.6045 (0.0014)	0.5086 (0.0016)	0.3734 (0.0017)
	Robust	0.6730 (0.0018)	0.6035 (0.0018)	0.5049 (0.0018)	0.3664 (0.0023)

CHAPTER 7

SUMMARY AND FUTURE WORK

7.1 Summary

This dissertation consists two parts. In the first part, which contains Chapter 2, Chapter 3 and Chapter 4, we focus on imbalanced data classification problems and proposed two new approaches within the pairwise learning framework.

The pairwise robust support vector machine automatically balances the impact of two imbalanced classes by pairing positive examples with negative ones. Simulations and real-world applications show that our new approach can effectively reduce the prediction error for the minority class. Compared with resampling techniques in the literature, it demonstrated the largest AUC. We also extended the PRSVM algorithm to handle non-linear classification problems by using the kernel trick. It demonstrates superiority over the traditional Kernel SVM approach on linearly inseparable data, especially when the data is highly imbalanced.

We applied the PRSVM model to fMRI based autism diagnosis. The performance is significantly better than some classical machine learning classifiers in terms of high autism patient detection and AUC score. Moreover, our findings reveal that the performance of the PRSVM model improves as the number of samples increases. This is significant because previous research shows traditional machine learning algorithms often struggled to perform well on large datasets. The ability of the PRSVM model to scale and maintain high performance with larger datasets is a valuable characteristic, making it more applicable in practical settings.

In the second part, which contains Chapter 5 and Chapter 6, we introduced corren-

tropy loss into deep neural networks to investigate the robustness of different neural networks. We proposed to implement robust deep neural networks using the correntropy loss and four two-stage algorithms. We observed that the robust deep neural networks outperformed traditional networks when dealing with data containing outliers or exhibiting skewness. This indicates that the incorporation of the correntropy loss played a significant role in improving the robustness of the network performance.

The superiority of two-stage algorithms is a serendipity. The original motivation of these algorithms is to study how the robust loss plays roles in the network construction process, not for better performance. The simulations surprisingly show that all two-stage algorithms are consistently better than their one-stage counterparts, regardless the loss function used.

We introduced robust loss into adversarial attacks, which involved training the network and generating adversarial examples. We conducted application experiments to evaluate the performance of this approach. The results of our experiments demonstrated certain advantages when using robust loss for both training the model and generating adversarial examples. The model trained with robust loss exhibited improved performance, and the generated adversarial examples demonstrated enhanced attack capability compared to other loss functions.

7.2 Future works

There are several open problems remaining for future research.

Firstly, in the PRSVM approach, we discovered through simulations that the parameter σ is not very sensitive, and a moderate choice typically yields satisfactory results. In our experiments, we fixed $\sigma = 1$ without tuning it for the best performance. However, we do not expect this generic choice will perform the best in all scenarios. It

would be worthwhile to develop tuning strategies that can enhance the performance of PRSVM.

Although PRSVM exhibited the largest AUC (Area Under the Curve) in all our experiments, its prediction errors, specifically the False Positive Rate (FPR) and the False Negative Rate (FNR), were not the smallest. It should be noted that AUC is independent of the intercept, while prediction errors are not. A plausible conjecture is that our estimation of the intercept is suboptimal, and a refined approach is expected to improve the prediction errors.

In our current research, we have incorporated the Gaussian kernel into the Kernel-PRSVM approach. However, in future studies, we have plans to explore the utilization of other kernel functions and evaluate their performance in comparison to the Gaussian kernel. By investigating different kernel functions, we aim to understand their impact on classification accuracy and explore potential improvements they may offer. Furthermore, similar to linear classifiers, we intend to explore alternative algorithms or techniques to estimate the intercept (b) in a more effective manner. This approach will allow us to achieve higher classification accuracy while still maintaining a high AUC value, providing better overall performance. By refining the intercept estimation process, we expect to enhance the effectiveness and robustness of the Kernel-PRSVM approach.

For both PRSVM and its kernel version, the superiority has only been verified by empirical studies. It would be interesting to explore their mathematical properties by conducting generalization analysis so we are able to theoretically explain their effectiveness. This would provide a deeper understanding of the underlying mechanisms of PRSVM and its performance in various scenarios.

For the robust deep learning study in Chapter 5, we have focused on the fully

connected deep forward neural network and the LSTM for regression analysis. CNN is another representative algorithm of deep learning and is particularly confidential for its excellent performance in image processing and computer vision. We can similarly develop robust convolutional neural networks. However, it seems CNN is more widely used in classification problems while the correntropy loss is more appropriate for regression analysis. We explored the idea of developing robust CNNs for classification problems and did not get superior results. But the idea of two-stage training is still promising and it would be interesting to develop two-stage CNN algorithms with appropriate classification loss functions, such as the cross entropy loss.

Although the use of robust loss does not generally improve the performance of CNNs, we see it may play roles in the adversarial attack and defense, especially for image classification, as shown in Chapter 7. However, the impact on complex three-channel images such as CIFAR-10 data seems less significant. This might be caused by network structure of selection of hyperparameters, if it is not an inherent characteristic of this data. More experiments with more complex network architectures or more appropriate hyperparameters may help to provide an affirmative answer to the questions. Additionally, it is also worth to apply the robust loss to the framework of attack-defense integration to train the neural network and analyze its robustness.

REFERENCES

[1] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *European conference on machine learning*, pages 39–50. Springer, 2004.

[2] Mohammad R Arbabshirani, Sergey Plis, Jing Sui, and Vince D Calhoun. Single subject prediction of brain disorders in neuroimaging: Promises and pitfalls. *Neuroimage*, 145:137–165, 2017.

[3] Patrick Bateson and Peter Gluckman. Plasticity and robustness in development and evolution. *International Journal of Epidemiology*, 41(1):219–223, 2012.

[4] JW Belliveau, DN Kennedy, RC McKinstry, BR Buchbinder, RMt Weisskoff, MS Cohen, JM Vevea, TJ Brady, and BR Rosen. Functional mapping of the human visual cortex by magnetic resonance imaging. *Science*, 254(5032):716–719, 1991.

[5] Xia-an Bi, Yang Wang, Qing Shu, Qi Sun, and Qian Xu. Classification of autism spectrum disorder using random support vector machine cluster. *Frontiers in genetics*, 9:18, 2018.

[6] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[7] Francisco J Castellanos, Jose J Valero-Mas, Jorge Calvo-Zaragoza, and Juan R Rico-Juan. Oversampling imbalanced data in the string space. *Pattern Recognition Letters*, 103:32–38, 2018.

[8] Badong Chen, Lei Xing, Haiquan Zhao, Nanning Zheng, and José C. Principe. Generalized correntropy for robust adaptive filtering. *IEEE Transactions on Signal Processing*, 64(13):3376–3387, 2016.

[9] Shuo Chen, Jian Kang, and Guoqing Wang. An empirical bayes normalization method for connectivity metrics in resting state fmri. *Frontiers in neuroscience*, 9:316, 2015.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[11] David D Cox and Robert L Savoy. Functional magnetic resonance imaging (fmri)"brain reading": detecting and classifying distributed patterns of fmri activity in human visual cortex. *Neuroimage*, 19(2):261–270, 2003.

[12] Cameron Craddock, Yassine Benhajali, Carlton Chu, Francois Chouinard, Alan Evans, András Jakab, Budhachandra Singh Khundrakpam, John David Lewis, Qingyang Li, Michael Milham, et al. The neuro bureau preprocessing initiative: open sharing of preprocessed neuroimaging data and derivatives. *Frontiers in Neuroinformatics*, 7:27, 2013.

[13] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[14] Bruno Légora Souza da Silva, Fernando Kentaro Inaba, Evandro Ottoni Teatini Salles, and Patrick Marques Ciarelli. Outlier robust extreme machine learning for multi-target regression. *Expert Systems with Applications*, 140:112877, 2020.

[15] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR, 2018.

[16] Tiago de Oliveira Nogueira, Gilderlânio Barbosa Alves Palacio, Fabrício Damasceno Braga, Pedro Paulo Nunes Maia, Elineudo Pinho de Moura, Carla Freitas de Andrade, and Paulo Alexandre Costa Rocha. Imbalance classification in a scaled-down wind turbine using radial basis function kernel and support vector machines. *Energy*, 238:122064, 2022.

[17] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[18] John E. Dennis Jr and Roy E. Welsch. Techniques for nonlinear least squares and robust regression. *Communications in Statistics-Simulation and Computation*, 7(4):345–359, 1978.

[19] John Doyle and Gunter Stein. Robustness with observers. *IEEE transactions on automatic control*, 24(4):607–611, 1979.

[20] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[21] Simon B Eickhoff, Klaas E Stephan, Hartmut Mohlberg, Christian Grefkes, Gereon R Fink, Katrin Amunts, and Karl Zilles. A new spm toolbox for com-

bining probabilistic cytoarchitectonic maps and functional imaging data. *Neuroimage*, 25(4):1325–1335, 2005.

[22] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

[23] Tom Fawcett. "in vivo" spam filtering: a challenge problem for kdd. *ACM SIGKDD Explorations Newsletter*, 5(2):140–148, 2003.

[24] Yunlong Feng, Jun Fan, and Johan A.K. Suykens. A statistical learning approach to modal regression. *Journal of Machine Learning Research*, 21(2):1–35, 2020.

[25] Yunlong Feng, Xiaolin Huang, Lei Shi, Yuning Yang, Johan AK Suykens, et al. Learning with the maximum correntropy criterion induced losses for regression. *J. Mach. Learn. Res.*, 16(30):993–1034, 2015.

[26] Yunlong Feng and Qiang Wu. Learning under $(1 + \epsilon)$-moment conditions. *Applied and Computational Harmonic Analysis*, 49(2):495–520, 2020.

[27] Yunlong Feng and Qiang Wu. A framework of learning through empirical gain maximization. *Neural Computation*, 33(6):1656–1697, 2021.

[28] Yunlong Feng, Yuning Yang, Xiaolin Huang, Siamak Mehrkanoon, and Johan AK Suykens. Robust support vector machines for classification with nonconvex and smooth losses. *Neural computation*, 28(6):1217–1247, 2016.

[29] Yunlong Feng and Yiming Ying. Learning with correntropy-induced losses for regression with mixture of symmetric stable noise. *Applied and Computational Harmonic Analysis*, 48(2):795–810, 2020.

[30] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[31] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

[32] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2011.

[33] Christophe Garcia and Manolis Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 26(11):1408–1423, 2004.

[34] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[35] Kaan Gokcesu and Hakan Gokcesu. Generalized huber loss for robust learning and its efficient minimization for a robust statistics. *arXiv preprint arXiv:2108.12627*, 2021.

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press Cambridge, 2016. `http://www.deeplearningbook.org`.

[37] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. Making machine learning robust against adversarial inputs. *Communications of the ACM*, 61(7):56–66, 2018.

[38] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[39] Marko Grobelnik. Feature selection for unbalanced class distribution and naive bayes. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 258–267. Citeseer, 1999.

[40] Xin Guo, Luisa F Polanía, and Kenneth E Barner. Group-level emotion recognition using deep models on image scene, faces, and skeletons. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 603–608, 2017.

[41] Deepak Gupta, Barenya Bikash Hazarika, and Mohanadhas Berlin. Robust regularized extreme learning machine with asymmetric huber loss function. *Neural Computing and Applications*, 32(16):12971–12998, 2020.

[42] Frank R Hampel, Elvezio M Ronchetti, Peter Rousseeuw, and Werner A Stahel. *Robust statistics: the approach based on influence functions*. Wiley-Interscience; New York, 1986.

[43] David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102, 1978.

[44] Anibal Sólon Heinsfeld, Alexandre Rosa Franco, R Cameron Craddock, Augusto Buchweitz, and Felipe Meneguzzi. Identification of autism spectrum disorder

using deep learning and the abide dataset. *NeuroImage: Clinical*, 17:16–23, 2018.

[45] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[46] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[47] Jin Huang and Charles X Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering*, 17(3):299–310, 2005.

[48] Xiaolin Huang, Lei Shi, and Johan AK Suykens. Support vector machine classifier with pinball loss. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):984–997, 2013.

[49] Peter J Huber. Robust regression: asymptotics, conjectures and monte carlo. *The annals of statistics*, pages 799–821, 1973.

[50] Peter J Huber. Robust statistics. In *International encyclopedia of statistical science*, pages 1248–1251. Springer, 2011.

[51] Tommi S Jaakkola and David Haussler. Probabilistic kernel regression models. In *Seventh International Workshop on Artificial Intelligence and Statistics*. PMLR, 1999.

[52] Brian Alan Johnson, Ryutaro Tateishi, and Nguyen Thanh Hoan. A hybrid pansharpening approach and multiscale object-based image analysis for map-

ping diseased pine and oak trees. *International journal of remote sensing*, 34(20):6969–6982, 2013.

[53] Leo Kanner et al. Autistic disturbances of affective contact. *Nervous child*, 2(3):217–250, 1943.

[54] Shwet Ketu and Pramod Kumar Mishra. Scalable kernel-based svm classification algorithm on imbalance air quality data for proficient healthcare. *Complex & Intelligent Systems*, 7(5):2597–2615, 2021.

[55] Konrad Paul Körding and Daniel M. Wolpert. The loss function of sensorimotor learning. *Proceedings of the National Academy of Sciences*, 101(26):9839–9842, 2004.

[56] Sotiris Kotsiantis, Dimitris Kanellopoulos, and Panayiotis Pintelas. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[57] Michał Koziarski. Radial-based undersampling for imbalanced data classification. *Pattern Recognition*, 102:107262, 2020.

[58] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[59] Tencent Keen Security Lab. Experimental security research of tesla autopilot. *Tencent Keen Security Lab*, 2019.

[60] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[61] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[62] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

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

[64] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010.

[65] Weichao Lin, Chih-Fong Tsai, YaHan Hu, and JingShang Jhang. Clustering-based undersampling in class-imbalanced data. *Information Sciences*, 409:17–26, 2017.

[66] Weifeng Liu, Puskal P. Pokharel, and Jose C. Principe. Correntropy: properties and applications in non-Gaussian signal processing. *IEEE Transactions on Signal Processing*, 55(11):5286–5298, 2007.

[67] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[68] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[69] Maciej A Mazurowski, Piotr A Habas, Jacek M Zurada, Joseph Y Lo, Jay A Baker, and Georgia D Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 21(2-3):427–436, 2008.

[70] Giovanna Menardi and Nicola Torelli. Training and assessing classification rules with imbalanced data. *Data mining and knowledge discovery*, 28(1):92–122, 2014.

[71] Gregory P Meyer. An alternative probabilistic interpretation of the huber loss. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 5261–5269, 2021.

[72] Anurag Mittal and Nikos Paragios. Motion-based background subtraction using adaptive kernel density estimation. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. Ieee, 2004.

[73] Wing WY Ng, Guangjun Zeng, Jiangjun Zhang, Daniel S Yeung, and Witold Pedrycz. Dual autoencoders features for imbalance classification problem. *Pattern Recognition*, 60:875–889, 2016.

[74] Seiji Ogawa, Tso-Ming Lee, Alan R Kay, and David W Tank. Brain magnetic resonance imaging with contrast dependent on blood oxygenation. *proceedings of the National Academy of Sciences*, 87(24):9868–9872, 1990.

[75] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos,

D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[76] Jose C. Principe. *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*. Springer Science & Business Media, 2010.

[77] Foster Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine learning*, 42(3):203–231, 2001.

[78] Shilin Qiu, Qihe Liu, Shijie Zhou, and Chunjiang Wu. Review of artificial intelligence adversarial attack and defense technologies. *Applied Sciences*, 9(5):909, 2019.

[79] Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5(53):2607, 2020.

[80] Fulong Ren, Peng Cao, Wei Li, Dazhe Zhao, and Osmar Zaiane. Ensemble based adaptive over-sampling method for imbalanced data learning in computer aided detection of microaneurysm. *Computerized Medical Imaging and Graphics*, 55:54–67, 2017.

[81] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[82] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*. John wiley & sons, 2005.

[83] Charles Smart Roy and Charles S Sherrington. On the regulation of the blood-supply of the brain. *The Journal of physiology*, 11(1-2):85, 1890.

[84] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[85] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[86] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[87] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[88] Bhaskar Sen, Gail A Bernstein, Tingting Xu, Bryon A Mueller, Mindy W Schreiner, Kathryn R Cullen, and Keshab K Parhi. Classification of obsessive-compulsive disorder from resting-state fmri. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3606–3609. IEEE, 2016.

[89] Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:1912.10944*, 2019.

[90] Lizhen Shao, Cong Fu, Yang You, and Dongmei Fu. Classification of asd based on fmri data with deep learning. *Cognitive Neurodynamics*, 15(6):961–974, 2021.

[91] Lizhen Shao, Yang You, Haipeng Du, and Dongmei Fu. Classification of adhd with fmri data and multi-objective optimization. *Computer Methods and Programs in Biomedicine*, 196:105676, 2020.

[92] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pages 1528–1540, 2016.

[93] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[94] Xiaotong Shen, George C Tseng, Xuegong Zhang, and Wing Hung Wong. On ψ-learning. *Journal of the American Statistical Association*, 98(463):724–734, 2003.

[95] Stephen M Smith, Mark Jenkinson, Mark W Woolrich, Christian F Beckmann, Timothy EJ Behrens, Heidi Johansen-Berg, Peter R Bannister, Marilena De Luca, Ivana Drobnjak, David E Flitney, et al. Advances in functional and structural mr image analysis and implementation as fsl. *Neuroimage*, 23:S208–S219, 2004.

[96] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14:199–222, 2004.

[97] Vicenc Soler, Jesus Cerquides, Josep Sabria, Jordi Roig, and Marta Prim. Imbalanced datasets classification by fuzzy rule extraction and genetic algorithms. In *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)*, pages 330–336. IEEE, 2006.

[98] Sutao Song, Zhichao Zhan, Zhiying Long, Jiacai Zhang, and Li Yao. Comparative study of svm methods combined with voxel selection for object category classification on fmri data. *PloS one*, 6(2):e17191, 2011.

[99] Fred A. Spiring. The reflected normal loss function. *Canadian Journal of Statistics*, 21(3):321–330, 1993.

[100] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[101] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[102] Mahbod Tavallaee, Natalia Stakhanova, and Ali Akbar Ghorbani. Toward credible evaluation of anomaly-based intrusion-detection methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(5):516–524, 2010.

[103] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.

[104] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, 2001.

[105] Philippe Van Kerm. Adaptive kernel density estimation. *The Stata Journal*, 3(2):148–156, 2003.

[106] Konstantinos Veropoulos, Colin Campbell, Nello Cristianini, et al. Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on AI*, volume 55, page 60. Stockholm, 1999.

[107] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

[108] Kefan Wang, Jing An, Zibo Yu, Xingshu Yin, and Chao Ma. Kernel local outlier factor-based fuzzy support vector machine for imbalanced classification. *Concurrency and Computation: Practice and Experience*, 33(13):e6235, 2021.

[109] Shan Wang, Feng Duan, and Mingxin Zhang. Convolution-gru based on independent component analysis for fmri analysis with small and imbalanced samples. *Applied Sciences*, 10(21):7465, 2020.

[110] Xinyue Wang, Jian Xu, Tieyong Zeng, and Liping Jing. Local distribution-based adaptive minority oversampling for imbalanced data classification. *Neurocomputing*, 422:200–213, 2021.

[111] Xueqin Wang, Yunlu Jiang, Mian Huang, and Heping Zhang. Robust variable selection with exponential squared loss. *Journal of the American Statistical Association*, 108(502):632–643, 2013.

[112] Gary M Weiss. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter*, 6(1):7–19, 2004.

[113] Tsui-Wei Weng, Huan Zhang, Pinyu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018.

[114] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International conference on machine learning*, pages 5286–5295. PMLR, 2018.

[115] Lingxi Xie, Richang Hong, Bo Zhang, and Qi Tian. Image classification and retrieval are one. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 3–10, 2015.

[116] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17:151–178, 2020.

[117] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *Journal of machine learning research*, 10(7), 2009.

[118] Xin Yang, Mohammad Samiul Islam, and AM Arefin Khaled. Functional connectivity magnetic resonance imaging classification of autism spectrum disorder using the multisite abide dataset. In *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 1–4. IEEE, 2019.

[119] Xin Yang, Paul T Schrader, and Ning Zhang. A deep neural network study of the abide repository on autism spectrum classification. *International Journal of Advanced Computer Science and Applications*, 11(4), 2020.

[120] L Yin, Q Wu, and D Hong. Statistical methods and software package for medical trend analysis in health rate review process. *J Health Med Inform*, 7(219), 2016.

[121] Qingbao Yu, Erik B Erhardt, Jing Sui, Yuhui Du, Hao He, Devon Hjelm, Mustafa S Cetin, Srinivas Rachakonda, Robyn L Miller, Godfrey Pearlson, et al. Assessing dynamic brain graphs of time-varying connectivity in fmri data: application to healthy controls and patients with schizophrenia. *Neuroimage*, 107:345–355, 2015.

[122] Olaf Zawacki-Richter, Victoria I Marín, Melissa Bond, and Franziska Gouverneur. Systematic review of research on artificial intelligence applications in higher education–where are the educators? *International Journal of Educational Technology in Higher Education*, 16(1):1–27, 2019.

[123] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied optics*, 29(32):4790–4797, 1990.

[124] Yun Zhang, Shiyu Yang, Hongbo Li, and ZhenHua Xu. Shadow tracking of moving target based on cnn for video sar system. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 4399–4402. IEEE, 2018.

[125] Xiaohu Zhao, Peijun Wang, Chunbo Li, Zhenghui Hu, Qian Xi, Wenyuan Wu, and Xiaowei Tang. Altered default mode network activity in patient with anxiety disorders: an fmri study. *European journal of radiology*, 63(3):373–378, 2007.