FEATURE REPRESENTATION FOR LARGE-SCALE DATA SET

YANXING HU

PhD

The Hong Kong Polytechnic University

2021

The Hong Kong Polytechnic University

Department of Computing

Feature Representation for Large-scale Data Set

Yanxing Hu

A thesis submitted in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

May 2019

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

. . . . . .

Yanxing Hu

Abstract

Feature representation is one of the most important research topics in Machine Learning (ML) area. In machine learning, representation of features means mapping the raw data into a new feature space that can be effectively exploited in machine learning tasks. Many supervised and unsupervised approaches, including supervised dictionary learning, Fuzzy and rough logics, Principal Component Analysis (PCA), local linear embedding, have been employed for feature representation of different types of data sets. The coming of the big data era brings both opportunities and challenges to the studies on feature representation. In real applications, the scale and the complexity of employed data far exceed the previous scenarios. On the one hand, the large volume of data set enables more complicate models be employed for feature representation, on the other hand, the multi-data source, complicate data structure and high computational requirement bring the new difficulties to the feature representation for huge data sets. In this study, concentrating on the feature representation problem for large-scale data set and related applications, new algorithms were proposed so that the obtained feature mapping enables better results for machine learning tasks.

Our study starts with the feature representation for data set with discrete values. For data sets with discrete values, the features often contain some categorical information about the data points. This study solves the feature representation of this kind of data by providing a novel rough set-based feature reduction approach, to efficiently and reliably extract the necessary information in the features while removing the redundant information of the data set.

Our second work is to provide a matrix decomposition based unsupervised pre-training approach for the feature representation. One of the important unsupervised feature representations approach is based on clustering models. However, clustering approaches are time-consuming, especially for large-scale data sets. An eigenvector based unsupervised pre-training approach is therefore proposed for feature representation, and combined as the first layer of the Radial Basis Function Neural Network(RBFNN).

Our third work concentrates on the feature representation for the data from multiple sources/views. A canonical correlation based-Auto encoder model is proposed for the feature fusion representation issue of the multi-domain data sets. The proposed model is consequently applied to the wind speed forecasting scenario to improve the wind speed forecasting accuracy.

Finally, we proposed a localize generalization error based data reduction approach, this approach can reliably reduce the training set for some large-scale data set, which provide a thought for the large-scale learning takes. This approach is highly related to the distribution of the values for each feature, it can be seen from this work that the representation of the features can affect the necessary number of training samples.

In summary, we make the following contributions: (i) algorithms and applications for feature representation on different types of large scale data sets; (ii) multi-domain feature fusion approach and applications; (iii) algorithms for computing the safe regions for the sum-optimal point notification problem.

Acknowledgements

First and foremost I would like to thank my supervisor Prof. Jane You and Dr. James N.K, Liu, for providing me with the amazing opportunity to do my Ph.D. under his supervision. their valuable advice, guidance, and support during the past few years has made me a better researcher and given me the practical skills necessary to succeed in my future career. Especially, in the most difficult time of my life, the last two years of my Ph.D. study, They gave me valuable support. I deeply admire their personal character.

A special thanks go out to my family, My Mom, Dad, and my wife, without whom I would not be where I am today. Their support and encouragement during both the good and the bad times have ensured that I always kept trying and succeeding. My father was diagnosed with gastric cancer in May of 2017. In the last two years of my Ph.D., my father has been struggling with the disease. He passed away finally. I really miss him.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Feature representation is one of the most fundamental topics in machine learning and pattern recognition. In machine learning applications, it usually employs an n-dimensional vector to give a numerical representation of an object, so that the information of the objects can be processed by the machine learning algorithms. In different scenarios, the values of the feature vector often represent different attributes of the represented objects. For example, when representing images, the feature values might correspond to the pixels of an image [1], when representing the health condition of a people, the feature values might be the terms in his medical examination report [2].

In machine learning applications, proper mapping of the raw features can effectively improve the performance of the employed learning algorithm. The significance of feature mapping is to find a new feature space for the reconstruction of the information representation. When the raw data is represented in the new feature space, the employed machine learning models may offer a better generalization, higher classification accuracy or lower Mean Square Error(MSE). That may greatly improve the efficiency and

performance of the employed learning algorithms [3].

Feature representation provides different benefits for machine learning applications [4]. For example, the raw features can be mapped to a lower-dimensional feature space, Such operation may remove the redundant information in the data set while keeping important features. Efficient feature reduction can not only obviously reduce the computational overhead, but also offer improvements to the results. Directly deleting less important features according to some given measure of feature importance from the raw data is the most intuitive approach for feature reduction. Some approaches can identify necessary data pattern information while lowering the number of features of the raw data set, e.g., Principal Component Analysis(PCA) can mapping the raw data to a lower-dimensional space while orthogonal basis set of new space shows the largest possible variance of the raw data set [5]. PCA is one of a linear feature representation approach to map the raw data to a lower-dimensional feature space, conversely, Manifold Learning approaches, such as Isometric feature mapping(ISOMAP) [6], map the raw data to a lower-dimensional feature space nonlinearly to detect some intrinsic properties of the raw data set. Besides mapping the raw data to a lower-dimensional feature space, mapping it to a higher-dimensional feature space can also offer benefits to machine learning applications. Linearly inseparable data may become linearly inseparable when it is mapped to a higher-dimensional feature space. In Support Vector Machine(SVM) [7], kennel functions are employed to map the raw data to a higher-dimensional feature space through the inner product to make the data linearly separable. In very recent studies, such as studies about deep neural networks [8], feature representation is proved can offer some comprehensive benefits of the employed machine learning models. e.g., Convolutional Neural Networks(CNNs) uses layers of convolution kernels to represent the features of raw images from different levels [9], by such

operation, the useless information in the images can be filtered, and the most discriminative features can be automatically extracted from the raw image.

Feature representation is so important that from the very beginning it is one of the most essential topics for machine learning studies. The development of feature representation methods has experienced three stages:

- **Manually feature selection by experienced domain experts.** Domain experts can manually select features according to their domain experience for specific application scenarios. The advantage of this approach is to use prior knowledge to improve the performance of the employed models. However, if the domain knowledge is not correct, or the given data sets contains a huge number of features and samples, selecting features manually may miss some useful information.

- **Feature selection or feature learning according to algorithms** Concentrating on the feature representation topic, previous researchers have proposed many algorithms. Generally speaking, these algorithms can be classified into two families. The supervised approaches, which learn the features according to the given labels of the used samples, and unsupervised approaches, which search the new feature space according to the relationship among the raw features. Feature selection algorithms are widely and successfully employed in machine learning and data mining studies. However, many feature representation algorithms were not Specifically proposed for large scale data sets, some modification of these approaches can be proposed for large scale data sets learning applications.

- **Feature learning by machine reasoning** Machine reasoning can learn some features that hardly identified by human experts and traditional algorithms. The hottest topic in the past ten years, multi-layer Neural Networks (NNs), well known

as deep learning, is the most effective approach for machine reasoning feature representation. Driven by the data itself, can extract different granularity of features in different levels of the reasoning model. The great success of deep learning, both in academia and industry, has proved the great advantages of this approach. The problem is that the family of deep learning approaches offer a great improvement on Computer Vision (CV) and Natural Language Processing (NLP) applications with a large number of training samples, for applications with large data set in other fields, the performance of deep learning seems not superior to other algorithms.

In general, feature representation can be divided into two main families of approaches, feature selection, and feature learning.

The objective of feature selection is to find the optimal subset of the given features of the processed data set. By feature selection, irrelevant features or redundant features can be removed, so that the number of features can be reduced to improve the precision of the employ learning models and save the computational resources. On the other hand, those truly necessary features are identified and given higher weight to make the learning tasks obtain better results.

The essence of feature selection is a combinatorial optimization problem of the features. Theoretically, the optimal subset of features can be obtained via exhaustive search strategy, the searching space for n features is 2^n, the amount of computation increases exponentially accompany with the increase of the dimensions. In real applications, the processed data set usually contains more than hundreds of features, therefore the exhaustive search method is not practical. Other searching strategies include the heuristic search method and the random search method [10, 11]. The target of feature selection

algorithms is to seek a better balance point between the computational efficiency and the quality of the obtained subset.

There is a close relationship between feature selection and the classification or regression algorithms. According to the evaluation criteria of the selected subsets and the combination of the subsequent learning algorithms, feature selection can be divided into three types: embedded, filtered and wrapper.

- **Embedded feature selection** In embedded feature selection, the feature selection algorithm itself is embedded into the learning algorithm as a component. The most typical embedded algorithm is the decision tree algorithm, e.g. ID3 [12], C4.5 [13], and CART [14]. A decision tree algorithm must select a feature in every recursive step of the tree growth process. So the process of decision tree generation is also the process of feature selection.
- **Filtered feature selection** The evaluation criteria of filtered feature selection is obtained from the processed data set rather than the employed learning algorithm. Filtered feature selection usually selects features with a high correlation with the labels to improve the precision of the classifier. The evaluation criteria for filtering feature selection can be divided into four categories: distance measurement, information measurement, correlation measurement, and consistency measurement. The most commonly used criteria include (1) cross-entropy [15], (2) Information Gain (IG) [16], (3) Mutual Information (MI) [17], (4) K-L divergence [18], etc.
- **Wrapper feature selection** Wrapper feature selection uses the performance of the learning algorithm to evaluate the quality of the feature subset [19]. Therefore, to evaluate an obtained feature subset, the wrapper method needs to train a classifier first to evaluate the feature subset according to the performance of the classi-

fier. Subsets obtained with wrappers approach usually offer higher classification precision than those obtained with other feature selection approaches. However, compared with filtered approaches, wrapper approaches have higher computational complexity. Especially for large scale data sets, wrapper approaches usually cost much longer training time. Another problem is that the subset obtained with a wrapper approach some times can only perform well for a certain classifier. The classification precision usually decreases when the classifier is changed to another.

In real applications, the processed data is often complex, redundant, and variable. Features selecting from the raw data are usually not enough to represent the essence of the problem. It is necessary to use algorithms to learn and extract new features, with a supervised algorithm or un-supervised algorithm to solve the problems in the learned feature space. Supervised feature learning algorithms include supervised dictionary learning [20], Neural Networks (NNs) [21] and multilayer perceptron(MLP). Unsupervised feature learning algorithms include unsupervised dictionary learning [22], PCA, Independent Component Analysis (ICA) [23], self-encoder, and clustering algorithm.

These algorithms can be roughly divided into several families according to the input variables, output variables, and optimization methods.

- **Matrix decomposition** Matrix decomposition approaches include PCA, ICA, Linear Discriminate Analysis (LDA), etc. The input of this family of algorithms is the raw data matrix of the given data set, denoted as $V_{m\times n}$, where m is the number of samples and n is the number of features, usually, the target of these algorithms is to find a mapping matrix $W_{m\times n'}$ to represent the data set into an n'-dimensional

feature space. $W_{m\times n'}$ can be calculated via matrix computation such as finding the eigenvalues and eigenvectors. Most of these algorithms are unsupervised, the correlation among features and the potential structure of data is considered more in these algorithms.

- **L_p-norm-based approach** The supervised dictionary learning and Least Absolute Shrinkage and Selection Operator(LASSO) learning the features by adding a L_p-norm term for regularization in the objective functions [24]. By optimizing the objective function, the raw data can be mapped into a sparse feature space. In these algorithms, both the implicit structure of input data and label information is employed to optimize the feature space.

- **NNs and deep learning** In this family of algorithms, the weighs matrix W_{ij}^{L} is to calculated for feature mapping. A typical NN is composed of interconnected neurons. In each neuron, a non-linear transformation is conducted by a non-linear function. The neurons are connected layer-by-layer by edges with the corresponding weights. The network defines the calculation rules and transfers data from the input layer to the output layer. The weights can be calculated via gradient descent or backpropagation. NNs can be used for feature learning because they can learn the representation of output in the hidden layer.

  The great success of deep learning stems from the greedy layer-wise pre-training of multilayer NNs [25]. In the deep learning architecture, the output of each intermediate layer can be regarded as a representation of the original input data. Each layer uses the representation generated in the previous layer as input and generates a new representation as output for the higher layer. The bottom of the input layer is the original data, while the final layer outputs the final low-dimensional features

or representations. Both supervised and unsupervised learning are employed in deep learning architecture. In general, deep learning architectures contain Restricted Boltzmann Machine (RBM) [26], Autoencoder, and different versions of CNNs [27]. As mentioned previously, Deep learning performs quite well for huge data set in CV and NLP domains, but for data sets from other fields, the successful cases are much less.

In this thesis, we focus on the feature representation for large scale data set. There is lack of clear definition about the conception 'large scale data set'. In this study, we define it by the PAC learning theroy: in a formal statistical learning framework, based on the *i.i.d. assumption*, set the real distribution as $\mathcal{D}$, given the hypothesis class $\mathcal{H}$, $\delta \in (0, 1)$ and $\epsilon > 0$, for a large scale data set, the number of sample m should satisfies$m \gg \frac{log(|\mathcal{H}|/\delta)}{\epsilon}$. A high-quality representation of features enables the employed learning models to become more precise. To achieve this, both the inner structure of the data sets and the relationship between feature representation approaches and the learning models should be taken into consideration. The ability of feature representation algorithms to deal with large scale data sets becomes more important with the increase in the volume of data. For larger-scale data sets, the implicit structure of the data set is more complex, more important information is hidden in the data sets. Most feature selection approaches and feature learning algorithms were proposed in an earlier time, although these methods have been proved can work well in smaller data set by lots of previous studies, whether these methods can take full advantage of the benefits of big data still needs further study. Some modification on these methods to enable them to perform better in larger-scale data set is necessary and worth to investigate.

Besides the precision, the efficiency of the algorithm should also be considered

more when processing large scale data set. As mentioned above, researchers make great efforts to find a better balance point between saving computational cost and finding the optimal solution all the time. This is more important in large scale data processing tasks. Some algorithms which are acceptable for smaller data set, can not be employed for larger-scale data set because of the time cost and limited computational resource. The time complexity and space complexity of some algorithms tend to increase exponentially with the increase of data volumes.

An unavoidable topic for feature representation of large scale data set is deep learning. As the most important and successful feature learning method, deep learning has brought profound changes to the field of CV and NLP field. Such a feature learning approach makes the feature representation for image/sound signal data and corpora data become much easier. When processing large volumes of data in other fields, e.g. weather information data or health information data, directly employ deep learning can usually not provide better performance than traditional data mining approaches. How to make use of the great power of deep learning methods on other types of data is also worth investigating.

The study in this thesis is to partly solve the mentioned problems of the feature representation for large scale data set. we started from the feature selection problem on data sets with discrete data, using a modified rough set approach, which is a heuristic algorithm, to calculate the subset of features. A new calculation of heuristic information is proposed to consider the uncertainty among the raw features. This provides a new balance point between the computational cost and the quality of the subset.

The family of NNs algorithm has become more popular and important recently for the applicability and generality. Therefore the feature representation by NNs is the

main emphasis of our study. Radial Basis Function Neural Network (RBFNN) is one of the most popular Feedforward Neural Network architectures. Compared with a typical three-layer NN, it can map the raw feature with Radial Basis Function (RBF) kernels. the centers of the RBF kernels are considered as representative prototypes of the data samples in the feature space, if the centers are chosen properly, the RBFNN model may offer high discriminative power and generalization. However, chosen the RBF center is usually the cost of training time, especially when the data size is large. We proposed a new algorithm for training RBFNN to learn the new features of large scale data set which can greatly reduce the training time while keeping the classification accuracy of the RBFNN.

We also trying to explorer the application of deep learning for other types of data. In this thesis, we propose a modified Auto Enoder(AE) model to deal with learning tasks using large time-series data sets from multiple domains.

In this thesis, a feature bases localize generalization error based data reduction approach is also given, this approach can reliably reduce the training set for some large-scale data set. This approach is highly related to the distribution of the values for each feature, it can be seen from this work that the representation of the features can affect the necessary number of training samples.

The rest of this thesis is organized as follows. Chapter 2 gives an overview of the existing work related to feature learning and feature representation.

Chapter 3 (based on [28] and [29])studies the feature reduction for data sets with discrete values. We illustrated with examples that the previously proposed reduct definitions may spoil the hidden classification ability of a knowledge system by ignoring certian essential attributes in some circumstances. Consequently, by proposing a new

β-consistent notion, we analysed the relationship between the structures of Decision Table (DT) and different definitions of reduct in VPRS model. Then we gave a new notion of β-complement reduction that can avoid the defects of reduct notions defined in previous literatures. We also supplied the method to obtain the β-complement reduct using a decision table splitting algorithm, and finally demonstrated the feasibility of our approach with sample instances.

Chapter 4 based on [30]) introduced an approach to quickly determine the RBF centers for an RBFNN model in feature learning with RBFNN. An eigenvector based clustering method is employed to calculate the RBF centers in the input feature space. RBF centers for the RBFNN model thus can be determined very quickly by calculating the principal components of the data matrix instead of the iterative calculation process of k-means clustering. After that, the connecting weights of the network can be easily obtained via either pseudo-inverse solution or the gradient descent algorithm. To evaluate the proposed approach, the performance of RBFNNs trained via different training schemes is compared in the experiments. It shows that the proposed method greatly reduces the training time of an RBFNN while allowing the RBFNN to attain a comparable accuracy result.

Chapter 5 (partly based on [31]) studies the fusion representation of features among data from different domain with deep NN. It aimed to utilize multi-layer neural network on the feature representation of univariate and cross-domain time series weather records. By training up the forecasting model in the represented feature space, the weather forecasting accuracy was expected to be enhanced. Specifically, the proposed work firstly explored the potential of the multi-layer neural network for the feature representation of the univariate time series weather data. Based on the findings, several models were consequently employed for the fusing representation of multi-domain

weather data so that the weather prediction accuracy could be enhanced in advance.

Chapter 6 studies a fact that feature representation cannot only affect the performance of employed computational models, but also related to other aspects of a certain machine learning applications, e.g., the number of necessary training samples. A novel data reduction approach that is highly related to the features of a date set is proposed.

Chapter 7 concludes the thesis.

Chapter 2

Literature Review

Feature representation is one of the most important research topics in machine learning. As mentioned in previous chapters, in machine learning, representation of features means mapping the raw data into a new feature space that can be effectively exploited in machine learning tasks. In most real-life applications, the most common usage of feature representation is simply to lower the number of dimensions to (1) extract or emphasize the useful information and remove the noises and (2) to reduce the computation burden and speed up the analysis. For image processing and natural language processing application, the multi-layer NN has become the dominating feature presentation approach.

2.1 Feature reduction

Feature reduction is to find a subset of the original feature collection of a data set. Motivated by identifying the key characteristics of a data set and saving the compu-

tational resources, researchers did lots of efforts to use as few features as possible to solve a certain machine learning problem while keeping the high performance of the employed intelligence model.

Many supervised and unsupervised approaches are proposed for solving the feature reduction problem for different applications with different types of data in earlier studies. Principal Component Analysis (PCA) may be the most widely used feature reduction algorithm in many real applications. It can be used for any type of data set including data mining and statistics applications in different fields.

Using statistical information to filter the feature set is the most basic feature reduction approach, especially for data sets with Discrete values [32]. The statistic information including the distance metrics, correlation, mutual information, and consistency metrics. The features are kept or removed according to the value of these criteria. Wrapping is another basic family of feature reduction approaches [33]. When using this kind of approach, different subsets of the feature collection are obtained according to some given method, and the one by which the employed model provides the best performance is chosen as the reduced feature collections.

Many data reduction algorithms are embedded in the employed classification or regression models of the machine learning application [34]. The family of decision trees is one of the most popular machine learning models for classification, it's also widely used to select a subset of features. Especially, the C4.5 and CART algorithms of decision trees can support the feature reduction of data sets with both continuous and discontinuous values.

Another important family of embedding data reduction approach is adding R-1 norm as a penalty term in the loss function of the employed computational model. Least

Absolute Shrinkage and Selection Operator (LASSO) [24], as the representative algorithm of this family of approaches, can make the coefficients of the less important features decay to 0, this is also considered as an adaptive sparse representation of the data set.

Neural network (NN) based algorithms are another important family of feature reduction approach. By reducing the number of hidden neurons of a certain hidden-layer to less than the original number of input features, the original features are linearly combined according to different weights and nonlinearly transformed to some other forms [35, 36]. One main superiority of NN based approaches is the university, which means they can be used for any type of applications and offer acceptable results, another great advantage of using NNs is considering the correlation relationship among features, both in linearly and nonlinearly. Decision tree-related approaches usually based on the assumption that the features involved in machine learning applications are independent with each other, but in the real world, things are often more complicated. The shortage of NN based methods is lacking interpretability after the linear combination and nonlinear transformation in the hidden neurons.

2.1.1 Multi-layer Neural network based feature representation

The family of Multi-layer NNs, well known as deep learning [8], is currently the most popular research topics in machine learning. For its unprecedented success in both academia and industry. It worth a section to review the related works of this family of feature representation algorithm.

The theoretical foundation of multi-layer NN is proposed in 1989 [37]. AS a universal approximator, a feedforward network with a single layer is sufficient to represent

any function, but the layer may be infeasibly large and may fail to learn and generalize correctly. In many circumstances, using deeper models can reduce the number of units required to represent the desired function and can reduce the amount of generalization error. Also in 1989, Yan Lecun verified the backpropagation(BP) applied to Handwritten Zip Code Recognition by using LeNet [38]. This proves that there are advantages to extracting local features and combining them into more advanced features (high-level features). By forcing hidden neurons to combine with local sources of information, it is easy to build such knowledge into a network. Since the key information can be hidden in features of a data set, using a feature detector to learn features from every input of a certain layer is a wise approach. Some features should be kept so that the detector in the next layer can learn more advance representations of the features.

Although the theoretical foundation of deep learning was given in the 1980s, it was not widely used before 2006 [39]. The complex architecture will lead the overfitting problem for smaller data set, and the Gradient vanishing problem of multi-layer architecture makes it hard to train. The first acknowledged success of the multi-layer architecture is the deep belief network (DBN) by Hinton in 2006. Hinton proposed a solution to the gradient vanishing problem in multi-layer NN training [26]: unsupervised pre-training in layer-wise initializes the weight firstly and then supervised fine-tuning, and in 2011, Rectified linear units (Relu) is proposed as the activation function for the non-linear representation of the features and proved can effectively suppress the Gradient vanishing problem [40].

Many famous structures were soon proposed after 2012. By using Relu, AlexNet was proposed that using an end-to-end training approach while using dropout to partly solve the overfitting problem [41]. Deep Residual Net was proposed in 2015 [42]. This network can solve the degradation problem by using an identity shortcut connection so

that the performance of the network can improve continuously while the architecture of the network becomes deeper and deeper. That means more advance representation of features can be learned.

Although multi-layer NN has obtained great success in the past ten years, there are still limitations. Applications with image data and corpus data can obtain great profit from feature representation by NN with deep architectures. However, little study about deep learning for applications with other types of data was reported. In real-life scenarios, the features are often in more complicated conditions, traditional methods are still widely used in many fields.

Chapter 3

Discrete Value Feature Selection

Discrete value is the basic type of feature values, however, currently, little work focuses on the feature learning problem of the discrete value feature. Discrete value is very common in some important applications, for example in bioinformatics, the sequencing analysis results of gene expression are usually in discrete form. In this chapter, the Rough Set (RS) is discussed as an efficient approach for feature learning of discrete value features.

Rouge Set (RS) theory has been developed dramatically since its introduction by Pawlak in 1982 [43]. It provides a formal methodology aiming at data analysis problems that involve uncertain, imprecise or incomplete information, and has had widespread success in many artificial intelligence research fields. However, when the given information contains some errors, such as missing information and classification abnormalities or the given Decision Table (DT) is derived from a relatively smaller data set, the obtained results of the classical RS model cannot always perform well and shows a poor generalization ability [44]. The Variable Precision Rough Set (VPRS) [44] model

was consequently proposed by ziarko in 1993. The VPRS is one of the most important extensions of the RS which has been proved to be capable of efficiently solving the mentioned disadvantage of the RS model [45, 46] . By introducing a threshold β, the standard inclusion relation in RS is extended to majority inclusion relation in VPRS and data patterns can be analyzed from the perspective of statistics.

For both classical RS model and VPRS model, one of the core problems is to find some particular subsets of the attributes collection. Attributes out of such subsets can be deemed as redundant and removed without causing deterioration of classification quality and inducing brief decision rules inherent in the given tables [47–49]. These subsets of attributes are called the reduct of a RS or VPRS model. By calculating the reduct, necessary attributes are identified and redundant information can be removed. For classical RS model, the definition of reduct is uncontroversial [50]. Accordingly, previous investigations about the reduct problem in RS model have mainly focused on the algorithm of calculating reduct [50, 51]. Different with the classical RS model, in VPRS model, the definition of the reduct has been revised many times in previous investigations. Ziarko defined the β-reduct first [44], but unfortunately, subsequent researches reported that decision rule conflict will occur when using Ziarko's reduct definition [52–54]. Then a new definition called β-distribution reduct in VPRS is proposed [47, 53]. Different with the β-reduct that only requires the consistency of classification quality degree between the original DT and the reduct, the β-distribution reduct is a more rigorous definition that can avoid the decision rule conflict by keeping β-positive regions of the original DT consistent.

This study presents a further investigation on the reduct problem in VPRS model. In our investigation, we focus on the limitation of the β-distribution reduct and try to develop the reduct definition by considering the consistent property of a certain DT in

VPRS model. Firstly, we use examples to demonstrate that although the β-distribution reduct can keep the decision rule consistent, it still has the risk to lose hidden information of the original DT and weaken the system's hidden classification ability. To deal with this problem, we extend the traditional consistent notion to a β-consistent notion for a DT in VPRS model, and analyze the relationship between the β-consistent notion and different definitions of reduct. Based on this analysis, a β-complete reduct notion is proposed. The β-complete reduct is shown not only can keep the decision rule consistent but also avoid the weakening of the system's hidden classification ability. We also give a decision table splitting algorithm for obtaining the proposed β-complement reduct. All of the notions and investigations in this chapter can help develop the VPRS model further.

In the next section, we briefly review the classical RS and VPRS models. In Section 3.2, we discuss the previous definitions of reduct in VPRS and show their limitations. The newly proposed β-consistent notion is introduced in Section 3.3; and in Section 3.4, we analyze the relationship between the β-consistent notion and different definition of reduct; We propose the β-complete reduct and present the algorithm in Section 3.5. The last section gives the conclusion and possible work in future.

3.1 Preliminaries

In this section, some fundamental knowledge about RS and VPRS will be introduced for convenience in presenting our investigation.

A DT is characterized by a 4-tuple $S = \langle U, A = C \cup D, V, f \rangle$, where $U = \{x_1, x_2, ...x_n\}$ denotes a nonempty finite set called universe, A is a nonempty finite set

of attributes that contains condition attribute set $C = \{c_1, c_2, \cdots c_k\}$ and decision attribute set $D = \{d_1, d_2, \cdots d_h\}$, where $C \cap D = \emptyset$. $V = \cup_{\alpha \in C \cap D}$ and V_α is the *domain* of the attributes α, notice that here the term 'α' can be either condition attributes or decision attributes . $f : U \times A \rightarrow V$ is a total function such that $f(x, \alpha) \in V_\alpha$ for every $x \in U$ and $\alpha \in A$, called an information function. e.g. $f(x_i, \alpha_j) = v$ means that in this DT, w.r.t.a certain attribute α_j, the element x_i has the value v.

Given an arbitrary non-empty subset $B \subseteq A$, an indiscernibility relation is defined as:

$$IND(B) = \{(x_i, x_j) \in U \times U \,\big|\, f(x_i, \alpha) = f(x_j, \alpha) \,, \forall \alpha \in B\} \tag{3.1}$$

$IND(B)$ partial U into a family of disjoint subsets $U/IND(B)$ called a quotient set of U:

$$U/IND(B) = \{\, [x]_B \,|\, x \in U\} \tag{3.2}$$

where $[x]_B$ denotes equivalence class determined by x w.r.t. $IND(B)$, i.e., $:[x]_B = \{\, y \in U |\, (y, x) \in IND(B)\}$

Then for a DT, we can define the equivalence classes determined by condition attribute set C and equivalence classes determined by the decision attribute set D respectively as follows:

$$\{C_i, i = 1, 2, \cdots, m\} = \{\, [x]_C \,\big|\, x \in U\} \tag{3.3}$$

$$\{D_j, j = 1, 2, \cdots, n\} = \{\, [x]_D \,|\, x \in U\} \tag{3.4}$$

Especially, D_j is called the decision class.

In classical RS model, For C, the lower and upper approximations of D_j can be respectively defined as:

$$\underline{C}(D_j) = \{x \,|[x]_C \subseteq D_j\} \tag{3.5}$$

$$\overline{C}(D_j) = \{x \,|[x]_C \cap D_j \neq \emptyset\} \tag{3.6}$$

and the positive region is defined as $POS_C(D_j) = \underline{C}(D_j)$

A VPRS model was proposed as an important extension of classical RS model, which gives a less rigour definition of the inclusion relation in eq.(5) and eq.(6) to make the classical RS model more fault tolerant [44,50]. In VPRS model, for a given precision parameter value $\beta \in (0.5, 1]$, we denote:

$$\underline{C}^{\beta}(D_j) = \bigcup \{x \in C_i \,|\omega(C_i, D_j) \geq \beta\} \tag{3.7}$$

$$\overline{C}^{\beta}(D_j) = \bigcup \{x \in C_i \,|\omega(C_i, D_j) > 1 - \beta\} \tag{3.8}$$

where ω is the inclusion degree function defined as:

$$\omega(X, Y) = \begin{cases} \frac{|X \cap Y|}{|X|}, |X| > 0, \\ 0, |X| = 0. \end{cases} \tag{3.9}$$

and the β-positive region of D_j w.r.t. C is $POS_C^{\beta}(D_j) = \underline{C}^{\beta}(D_j)$.

Since $\beta > 0.5$, the less rigour inclusion relation in VPRS model is called the majority inclusion relation that is seen as the heart of the VPRS model.

When $\beta = 1$, the VPRS model is equal to a classical RS mode [44].

A notion called the classification quality degree (or called the degree of dependence of decision attribute set D w.r.t. condition attribute set C) is applied to measure the classification quality in classical RS model. If the decision attribute set D divides U into n decision classes, the classification quality degree w.r.t a certain attribute set C is:

$$\sigma_C(D) = \frac{\left| \sum_{j=1}^{n} \underline{C}(D_j) \right|}{|U|} \tag{3.10}$$

Similarly, in VPRS model, when a precision parameter β is given, the classification quality degree is:

$$\sigma_C^{\beta}(D) = \frac{\left| \sum_{j=1}^{n} \underline{C}^{\beta}(D_j) \right|}{|U|} \tag{3.11}$$

Positive regions and classification quality degree reflect the knowledge of a certain information system from the perspective of quality and quantity respectively [44, 54].

3.2 Attributes reduction in VPRS model

This section briefly introduces the previous definitions of *reduct* in VPRS model, and more importantly, we will show the limitation of the presently used β-distribution reduct in this section.

3.2.1 The β-reduct and its limitation in VPRS model

The quality of classification is often used to measure the classification ability of a DT. Nevertheless, the final rule inference is based on the concrete objects in positive

region. In classical RS model, the monotonicity of classification quality and positive region are uniform during the procedures of attribute reduction [55]. Accordingly, the same classification quality degree implies the consistency of positive region in classical RS model [50]. However, in VPRS model, these monotonicity properties will not be satisfied any more, therefore the same classification quality may result in different β-positive regions [54]. i.e., the final decision rules extracted from the reduct maybe in conflict with those extracted from the original DT under the same classification quality degree [52, 53, 55].

Definition 1: For a DT $S = (U, A = C \cup D, V, f)$ in VPRS model with precision parameter β, $B \subseteq C$, if:

$$\forall \alpha \in B,\ \sigma^{\beta}_{B-\{\alpha\}} \neq \sigma^{\beta}_{C} \tag{3.12}$$

$$\sigma^{\beta}_{B}(D) = \sigma^{\beta}_{C}(D) \tag{3.13}$$

such a subset B is called a β-reduct of C under the precision β, defined by Ziarko [44].

However, under some circumstances, the β-reduct, based on the equalization of classification quality degree, have to face the inconsistent problem. Some conflicts will be generated between the decision rules extracted from the original DT and the decision rules extracted from the obtained reduct. An example is shown in Table 3.1.

In sample DT 1, $o_1, o_2, \ldots, o_7$ are the elements in the universe, $C = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ is the condition attribute set of this DT, and $D = \{d\}$ is the decision attribute set. The values in the table are the attribute values of the corresponding elements. We have:

$D_1 = \{o_1, o_2, o_3\}$,

$D_2 = \{o_4, o_5, o_6, o_7\}$,

Table 3.1. Sample DT 1

U	α_1	α_2	α_3	α_4	d
o_1	0	1	1	1	1
o_2	1	1	1	1	1
o_3	1	1	0	0	1
o_4	0	1	0	1	0
o_5	1	1	1	1	0
o_6	1	0	1	0	0
o_7	1	1	1	1	0

Table 3.2. β-reduct DT of sample DT 1

U	α_2	α_4	d
o_1	1	1	1
o_2	1	1	1
o_3	1	0	1
o_4	1	1	0
o_5	1	1	0
o_6	0	0	0
o_7	1	1	0

as decision classes determined by d; and:

$C_1 = \{o_1\}$,

$C_2 = \{o_2, o_5, o_7\}$,

$C_3 = \{o_3\}$,

$C_4 = \{o_4\}$,

$C_5 = \{o_6\}$,

are equivalent classes determined by C.

Setting $\beta = 0.59$, we can see that $B = \{\alpha_2, \alpha_4\}$ is a β-reduct of sample DT 1 since it is easy to calculate $\sigma_B^{0.59}(D) = \sigma_C^{0.59}(D) = 1$. The DT in Table 3.2 is the β-reduct DT of the sample DT 1 in Table 3.1.

Although the classification quality degree values of the two DT are equal, from the β-reduct DT in table 3.2, we can obtain:

$POS^{0.59}{}_C(D_1) = \underline{C}^{0.59}(D_1) = \{o_1, o_3\}$,

$POS^{0.59}{}_C(D_2) = \underline{C}^{0.59}(D_2) = \{o_2, o_5, o_7, o_4, o_6\}$,

and:

$POS^{0.59}{}_B(D_1) = \underline{B}^{0.59}(D_1) = \{o_3\}$,

$POS^{0.59}{}_B(D_2) = \underline{B}^{0.59}(D_2) = \{o_1, o_2, o_4, o_5, o_7, o_6\}$,

which show that the β-positive regions are changed in the reduct $B = \{\alpha_2, \alpha_4\}$.

Specifically, from the obtained β-reduct $B = \{\alpha_2, \alpha_4\}$, we can extract the following decision rule: $(\alpha_2, 1) \wedge (\alpha_4, 1) \rightarrow (d, 0)$, which is supported by o_1, o_2, o_4, o_5 and o_7, when $\beta = 0.59$.

This is in conflict with the decision rule in the original DT: $(\alpha_1, 0) \wedge (\alpha_2, 1) \wedge (\alpha_3, 1) \wedge (\alpha_4, 1) \rightarrow (d, 1)$, supported by o_1, when $\beta = 0.59$. So we can see that the β-reduct may generate conflict classifcation information with the original DT.

3.2.2 The β-distribution reduct and its limitation

Since the β-reduct may lead to conflict problems, a β-distribution reduct has been proposed by Mi in subsequent research and widely used currently [53].

Definition 2: Given a DT $S = (U, A = C \cup D, V, f)$ in VPRS model with precision parameter β, $B \subseteq C$, B is a reduct of C if:

$$\forall \alpha \in B,\ L_C^{\beta} \neq L_{B-\{\alpha\}}^{\beta} \tag{3.14}$$

Table 3.3. Sample DT 2

U	α_1	α_2	α_3	α_4	α_5	d
x_1	1	1	1	1	1	1
x_2	1	1	0	1	1	1
x_3	0	0	1	0	0	1
x_4	1	1	2	1	1	1
x_5	1	1	0	1	0	2
x_6	1	1	0	1	1	2
x_7	0	0	1	2	1	2
x_8	1	1	0	1	1	2
x_9	1	1	2	1	1	2
x_{10}	1	0	2	1	1	3
x_{11}	1	0	2	1	1	4

$$L_B^\beta = L_C^\beta \tag{3.15}$$

where L is used to denote the collection of β-positive regions, $L_B^\beta = (\underline{B}^\beta(D_1), \underline{B}^\beta(D_2), \cdots, \underline{B}^\beta(D_n))$ and $L_C^\beta = (\underline{C}^\beta(D_1), \underline{C}^\beta(D_2), \cdots, \underline{C}^\beta(D_n))$.

From the definition, in VPRS, a β-distribution reduct is also a β-reduct, because that according to eq.(11), if eq.(15) is true, eq.(13) must be true. but conversely, a β-reduct is not definitely a β-distribution reduct: in the sample DT 1, $B = \{\alpha_2, \alpha_4\}$ is a β-reduct, but not a β-distribution reduct.

Obviously, the β-distribution reduct is a more rigorous definition than the Ziarko's definition and can ensure the consistency of the β-positive regions after the reduction. However, in our investigation, we find that the β-distribution reduct also has limitations. The sample DT 2 in Table 3.3 is an example.

In sample DT 2, set $\beta = 0.6$, we have:

$D_1 = \{x_1, x_2, x_3, x_4\}$,

$D_2 = \{x_5, x_6, x_7, x_8, x_9\}$,

$D_3 = \{x_{10}\}$,

$D_4 = \{x_{11}\}$,

as decision classes. In the meanwhile, for condition attributes set C we have equivalent classes:

$C_1 = \{x_1\}$,

$C_2 = \{x_2, x_6, x_8\}$,

$C_3 = \{x_3\}$,

$C_4 = \{x_4, x_9\}$,

$C_5 = \{x_5\}$,

$C_6 = \{x_7\}$

$C_7 = \{x_{10}, x_{11}\}$.

For $B = \{\alpha_3, \alpha_4\}$, we can obtain equivalent classes:

$B_1 = \{x_1\}$,

$B_2 = \{x_2, x_5, x_6, x_8\}$,

$B_3 = \{x_3\}$,

$B_4 = \{x_4, x_9, x_{10}, x_{11}\}$,

$B_5 = \{x_7\}$.

Consequently, it is easy to calculate that:

$POS^{0.6}{}_B(D_1) = \underline{B}^{0.6}(D_1) = \{x_1, x_3\}$,

$POS^{0.6}{}_B(D_2) = \underline{B}^{0.6}(D_2) = \{x_2, x_5, x_7, x_6, x_8\}$,

$POS^{0.6}{}_B(D_3) = \underline{B}^{0.6}(D_3) = \emptyset$,

$POS^{0.6}{}_B(D_4) = \underline{B}^{0.6}(D_4) = \emptyset$,

in the meanwhile, we also have:

$POS^{0.6}{}_C(D_1) = \underline{C}^{0.6}(D_1) = \{x_1, x_3\}$,

$POS^{0.6}{}_C(D_2) = \underline{C}^{0.6}(D_2) = \{x_2, x_5, x_7, x_6, x_8\}$,

$POS^{0.6}{}_C(D_3) = \underline{C}^{0.6}(D_3) = \emptyset$,

$POS^{0.6}{}_C(D_4) = \underline{C}^{0.6}(D_4) = \emptyset$.

So we have $L_B^\beta = L_C^\beta$. Accordingly, $B = \{\alpha_{3,}\alpha_4\}$ is a β-distribution reduct of C when $\beta = 0.6$.

Now the problem comes. Observing the original DT in Table 3.3, when $\beta = 0.6$, we can extract two special decision rules:

$(\alpha_1, 1) \wedge (\alpha_2, 0) \wedge (\alpha_3, 2) \wedge (\alpha_4, 1) \wedge (\alpha_5, 1) \rightarrow (d, (1or2))$, supported by x_4 and x_9;

$(\alpha_1, 1) \wedge (\alpha_2, 0) \wedge (\alpha_3, 2) \wedge (\alpha_4, 1) \wedge (\alpha_5, 1) \rightarrow (d, (3or4))$, supported by x_{10} and x_{11};

After the attributes reduction, we can extract:

$(\alpha_3, 2) \wedge (\alpha_4, 1) \rightarrow (d, (1or2or3or4))$, supported by x_4, x_9, x_{10} and x_{11}.

When in a certain decision rule, the decision attribute can get varied values, we say the conflict happens in the decision part of the decision rule. In this chapter, for convenience in presentation, we denote such conflict decision part of the decision rule as $(d, \emptyset)$.

In both of the original DT and the β-distribution reduct, when $\beta = 0.6$, x_4, x_9, x_{10} and x_{11} all support that $(\alpha_3, 2) \wedge (\alpha_4, 1) \rightarrow (d, \emptyset)$. Specifically, in the original DT, we can observe that when $f(x_i, \alpha_2) = 0$, there is $(\alpha_3, 2) \wedge (\alpha_4, 1) \rightarrow (d, (1or2))$, and when $f(x_i, \alpha_2) = 1$, the decision rule is $(\alpha_3, 2) \wedge (\alpha_4, 1) \rightarrow (d, (3or4))$, that means the α_2 cannot be simply considered as a redundant attribute, it has some hidden classification

ability that the attributes in the β-distribution reduct don't have. But in the β-distribution reduct, the attributes α_2 is excluded.

The problem is generated in the process of calculating the β-distribution reduct: both of the two conditions, $(\alpha_3, 2) \wedge (\alpha_4, 1) \rightarrow (d, (1or2))$ and $(\alpha_3, 2) \wedge (\alpha_4, 1) \rightarrow (d, (3or4))$, are considered as $(\alpha_3, 2) \wedge (\alpha_4, 1) \rightarrow (d, \emptyset)$ in general, because the value of d is in conflict. But actually, they are different because when $f(x_i, \alpha_2) = 1$, x_i with $(\alpha_3, 2) \wedge (\alpha_4, 1)$ has 50% probability to be classified in D_1 and 50% probability to be classified in D_2, and when $f(x_i, \alpha_2) = 0$, this sample point with $(\alpha_3, 2) \wedge (\alpha_4, 1)$ has 50% probability to be classified in D_3 and 50% probability to be classified in D_4. Accordingly, α_2 also contains necessary classification information of the DT, it should NOT be considered as a redundant attribute and removed from the β-distribution reduct.

From the above analysis, we can see that although the β-distribution reduct is a more rigorous definition than the β-reduct and can make sure the consistency of the β-positive regions after the reduction in order to avoid the decision rule conflict, it still has the limitations that it may exclude some necessary attributes, e.g. α_2 in sample DT 2. This limitation is generated due to those sample points with conflict decision rules. These samples are all considered as with a conflict decision rule $(d, \emptyset)$, and the β-distribution reduct thus ignores the differences among different conflict conditions. This may lead to the loss of some hidden classification information or knowledge in the original information system and weaken the system's classification ability.

3.3 β-consistent notion for a DT in VPRS

We have illustrated with examples that, in VPRS model, the current definitions of reduct all have limitations. For further investigation about the reduct in VPRS model, in this section, a β-consistent notion for a DT in VPRS is proposed first.

3.3.1 The consistent DT

In classical RS model, given a DT $S = (U, A = C \cup D, V, f)$, $f : U \times A \rightarrow V$ is the information function such that $f(x, \alpha) \in V_\alpha$ for every $x \in U$ and $\alpha \in A$ as defined in Section II.

Definition 3: A DT is a consistent DT [56] if $\forall x, y \in U$ where $x \neq y$, we have:

$$f(x, C) = f(y, C) \Rightarrow f(x, D) = f(y, D). \tag{3.16}$$

This consistency property can be also represented from the indiscernibility relation aspect: given a DT $S = (U, A = C \cup D, V, f)$, we say this DT is consistent if:

$$IND(C) \subseteq IND(D). \tag{3.17}$$

However, in VPRS model, because the majority inclusion relation is taken into consideration, determining requirement of whether a certain DT is consistent becomes complicated. Some initially inconsistent DT in RS model can be re-evaluated in VPRS model. Under VPRS models with different β values, a certain DT may change its consistency property. In the next section of this chapter, a new notion called β-consistency is proposed for VPRS model.

3.3.2 The β-consistent DT

Given a DT $S = (U, A = C \cup D, V, f)$ and a precision parameter β, $\forall x, y \in U, x \neq y$, for a certain equivalent class C_i, if $x \in C_i$, $y \in C_i$, $\exists D_j$ satisfies that $x \in POS_C^{\beta}(D_j)$, $y \in POS_C^{\beta}(D_j)$, then we define this DT is β-consistent under the precision β.

Definition 4: A DT is a β-consistent DT in VPRS model if $\forall x, y \in U$ where $x \neq y$, we have:

$$f(x, C) = f(y, C) \Rightarrow x, y \in POS_C^{\beta}(D_j) \tag{3.18}$$

Based on this definition, while given a precision parameter β , some inconsistent DT in RS becomes β-consistent in VPRS. The sample DT 1 in Table 3.1 is an example, obviously, it is an inconsistent DT: for sample points o_2 and o_5, we have $o_2 \in C_2$ and $o_5 \in C_2$, i.e., $f(o_2, C) = f(o_5, C)$, but in the meanwhile, $o_2 \in D_1$, $o_5 \in D_2$, i.e., $f(o_2, D) \neq f(o_5, D)$, which is in conflict with eq.16, so the DT in Table 3.1 is not consistent.

However,when we analyze the same DT in VPRS model, setting a precision parameter $\beta = 0.59$, we have equivalent classes:

$C_1 = \{o_1\}$,

$C_2 = \{o_2, o_5, o_7\}$,

$C_3 = \{o_3\}$,

$C_4 = \{o_4\}$,

$C_5 = \{o_6\}$,

and decision classes:

$D_1 = \{o_1, o_2, o_3\}$,

$D_2 = \{o_4, o_5, o_6, o_7\}$.

So we have:

$$POS_C^{0.59}(D_1) = \underline{C}^{0.59}(D_1) = \{o_1, o_3\},$$
$$POS_C^{0.59}(D_2) = \underline{C}^{0.59}(D_2) = \{o_2, o_5, o_7, o_4, o_6\}.$$

We can observe that, for any two elements in U of sample DT 1, if they belong to the same equivalent class C_i, we can find a decision class D_j, and both of these two elements belong to $POS_C^{0.59}(D_j)$. So, this DT is β-consistent when $\beta = 0.59$ in VPRS model.

Moreover, for the same DT in Table 3.1, if we set $\beta = 0.67$, the β-consistent property will change. When $\beta = 0.67$, the β-positive region of D_1 becomes:

$$POS_C^{0.67}(D_1) = \{o_1\},$$

and the β-positive region of D_2 is :

$$POS_C^{0.67}(D_2) = \{o_4, o_6, o_7\},$$

we will observe that the elements o_2, o_5 and o_7 in C_2 cannot be included by positive region of any D_j since that $\omega(C_2, D_1) = 2/3 < 0.67$ and $\omega(C_2, D_2) = 1/3 < 0.67$, i.e., for o_2, o_5 and o_7, we cannot find a decision class D_j to satisfy the requirement in definition 4. So, when $\beta = 0.67$, this DT is **NOT** a β-consistent DT (or β-inconsistent DT for presentation convenience).

From the indiscernibility relation aspect, we say a DT is β-consistent meaning that $IND(C)$ and $IND(D)$ are not in conflict when the classification precision is not higher than β for this DT in VPRS model, and from the above instance, we can see that the same DT may change its β-consistent property when the value of β is changed.

Theorem 1: for two precision parameters $\beta_1 \in (0.5, 1]$ and $\beta_2 \in (0.5, 1]$, set $\beta_2 \leq \beta_1$, if a certain DT is β_1-consistent, it must be a β_2-consistent DT.

Proof: from the definition, if a DT is β_1-consistent, that means all the elements in

each C_i can be classified into a β_1-positive region of a certain D_j, from the definition of β-positive region, we may obtain that $\omega(C_i, D_j) = (\left|C_i \cap D_j\right| / |C_i|) \geq \beta_1$, since $\beta_2 \leq \beta_1$, we can also obtain $\omega(C_i, D_j) \geq \beta_2$. Accordingly, any element in C_i also can be classified into β_2-positive region of a certain D_j. Thus this DT is also β_2-consistent.

Moreover, from the definition, the 1-consistent notion in VPRS is equal to the classical consistent notion. So the β-consistent notion can be considered as an extension of classical consistent notion, and we can easily get the following deduction:

Deduction 1: a consistent DT must be a β-consistent DT, $\forall\beta \in (0.5, 1]$.

For convenience in presentation of the following part, we firstly define other two notions:

$$G_C^{\beta}(x) = \{D_j \left| x \in POS_C^{\beta}(D_j) \right.\}, x \in U \tag{3.19}$$

$$R^{\beta}(C_i) = \{G_C^{\beta}(x) \,| x \in C_i\} \tag{3.20}$$

From the definition, the elements in collection $R^{\beta}(C_i)$ are constituted by the decision classes. For each C_i, if any element in it also belongs to the β-positive region of a certain decision class D_j, we put the decision class D_j into a corresponding collection $R^{\beta}(C_i)$.

Theorem 2: when $\beta \in (0.5, 1]$, $\left|R^{\beta}(C_i)\right| \in \{0, 1\}$, in other words, $\left|R^{\beta}(C_i)\right| < 2$.

Proof: It easily to find the example of $\left|R^{\beta}(C_i)\right| = 0$ and $\left|R^{\beta}(C_i)\right| = 1$. Observe the DT in Table 3.1, Set $\beta = 0.6$, we can get $R^{0.6}(C_{\{o_1\}}) = \{D_1\}$, so $\left|R^{0.6}(C_{\{o_1\}})\right| = 1$, and $R^{0.6}(C_{\{o_2,o_3\}}) = \emptyset$, so $\left|R^{0.6}(C_{\{o_2,o_3\}})\right| = 0$.

Denote $|C_i| = N$, if $\left|R^{\beta}(C_i)\right| = 2$, setting $R(C_i) = (D_{j'}, D_{j''})$, where $D_{j'} \neq D_{j''}$.

From eq. (4), we have $D_{j'} \cap D_{j''} = \emptyset$. Setting the number of elements in C_i belonging to $D_{j'}$ as m and the number of elements in C_i belonging to $D_{j''}$ as n, and, according to the definition of β-positive region, $m/N \geq \beta$, and $n/N \geq \beta$, $0.5 < \beta \leq 1$, so $m/N > 0.5$, $n/N > 0.5$, then we have $(m+n)/N > 1$, this is in conflict with $m + n = N$. The condition is similar when $\left|R^{\beta}(C_i)\right| = 3, 4, ...\infty$.

Theorem 2 shows that when the classification precision is larger than 0.5, one sample point can only support one decision rule.

From the definition of $R^{\beta}(C_i)$, we can see that when given a certain β, if a DT is β-consistent, $\forall C_i$, $\left|R^{\beta}(C_i)\right| = 1$; in the meanwhile, if in a DT, $\forall C_i$, we have $\left|R^{\beta}(C_i)\right| = 1$, this DT is β-consistent. Accordingly, in a DT:

$$\left|R^{\beta}(C_i)\right| = 1, \forall C_i \tag{3.21}$$

is the sufficient and necessary condition for that a DT is β-consistent. If in a DT, $\exists C_i$, where $\left|R^{\beta}(C_i)\right| = 0$, we can deduce that this DT is β-inconsistent, and the equivalent class C_i with $\left|R^{\beta}(C_i)\right| = 0$ is called *undecidable equivalent class* of this DT in our investigation. Moreover, the number of undecidable equivalent class shows the number of decision conflict conditions in the decision rules of a DT.

3.4 The relationships between the reducts and β-consistent property of a DT in VPRS model

In the previous sections, we have already discussed different definitions of attribute reduct in VPRS model: the β-reduct has already been proved to have limitations in

previous investigations; moreover, our example in Table 3.3 has illustrated that the β-distribution reduct also has limitations. Consequently, we proposed the definition of β-consistent property of a DT in VPRS model, which is an extension of the consistent notion in classical RS model. In this section, we will analyze the relationship between different definitions of reduct and the β-consistent property of a certain DT in VPRS.

If a DT is 1-consistent, the β-reduct is equal to the β-distribution reduct, where $\beta \in (0.5, 1]$.

If a DT is 1-consistent, that means for each equivalent class C_i, $\exists D_j$, where $(C_i \cap D_j) \big/ D_j = 1$, in the meanwhile, in Theorem 2, we have proved that when $\beta \in (0.5, 1]$, any equivalent class can be classified into one decision class at most ($\left|R^{\beta}(C_i)\right| < 2$). Therefore, $\forall \beta \in (0.5, 1)$, $POS_C^{\beta}(D_j) = POS_C^{1}(D_j)$. So the VPRS model becomes a classical RS model when the DT is 1-consistent.

In a classical RS model, as we mentioned above, the monotonicity of classification quality and positive regions are uniform during the procedures of attributes reduction. The same classification quality degree means the consistency of positive regions [50,55]. Accordingly, the β-reduct is equal to the β-distribution reduct, and the decision rule conflict can be avoided in the reduction process.

If an inconsistent DT is β-consistent for $\beta \in (0.5, 1)$, the β-reduct may lead to the decision rule conflict in VPRS model, while the β-distribution reduct can keep the decision rule consistent with those in the original DT. Moreover, the β-distribution reduct can also avoid the problem of loss hidden classification information.

If a DT is inconsistent but β-consistent where $\beta \neq 1$, that means there are decision conflicts in this DT initially, but these decision conflicts can be eliminated in VPRS

model when the classification precision requirement is lower than β as the majority inclusion relation is less rigour than the standard inclusion relation.

For example, in Table 3.1, the sample DT 1, a decision rule conflict is generated by o_2, o_5 and o_7:

$(\alpha_1, 1) \wedge (\alpha_2, 1) \wedge (\alpha_3, 1) \wedge (\alpha_4, 1) \rightarrow (d, 1)$, supported by o_2;

$(\alpha_1, 1) \wedge (\alpha_2, 1) \wedge (\alpha_3, 1) \wedge (\alpha_4, 1) \rightarrow (d, 0)$, supported by o_5 and o_7.

This DT is obviously a inconsistent DT in RS model, But in VPRS model, when we introduce $\beta = 0.59$, since two of the three sample points have supported $(d, 0)$, so the classification precision is $2/3 \geq 0.59$, the majority inclusion relation requirement is satisfied, which means that in this conflict, the majority (o_5 and o_7) of the three samples support the latter decision rule. So this conflict can be eliminated and we use the decision rule:$(\alpha_1, 1) \wedge (\alpha_2, 1) \wedge (\alpha_3, 1) \wedge (\alpha_4, 1) \rightarrow (d, 0)$ in VPRS model. Further, in general, for a originally inconsistent DT, if it is β-consistent where $\beta \in (0.5, 1)$, all the decision conflicts can be eliminated under a certain classification precision β like this.

For such DT, as we discussed in Part A of Section III, the decision rules extracted from the original DT may be in conflict with the decision rule extracted from the β-reduct under some conditions. Therefore, for such DT, only calculating the β-reduct is not enough.

On the other hand, for a β-consistent DT, the β-distribution reduct can not only keep the classification quality degrees consistent but also keep the decision rules consistent with the original DT: decision rules of a DT is determined by the positive regions [55, 57], the β-distribution reduct requires the β-positive regions to be kept con-

sistent after the attributes reduction, so the decision rules are also kept consistent in the β-reduct.

Additionally, if a DT is β-consistent, the β-distribution reduct of this DT can avoid the risk of losing hidden information which we mentioned in Part B, Section III. As we discussed above, the loss of hidden information is generated by neglecting the differences among different decision conflicts. From the definition of β-consistent, in a β-consistent DT, for each equivalent class C_i, $\exists D_j$, where $C_i \subseteq POS_C^{\beta}(D_j)$. Thus, $\forall C_i$, a definite decision rule: $\bigwedge(c, f(C_i, c)) \rightarrow (d, f(D_j, d))$. without any decision conflict (e.g. the $(d, \emptyset)$ condition in sample DT 2) can be extracted. Therefore the β-distribution reduct will not lose necessary hidden classification information or knowledge in the DT.

If a DT is β-inconsistent, $\beta \in (0.5, 1]$, use N to denote the number of undecidable equivalent classes, if $N = 1$, the β-distribution reduct can keep the decision rule consistent and will not lose information in the original DT.

As we discussed in previous section, the number of undecidable equivalent classes reflects the number of decision conflict conditions of a DT. in the meanwhile, we have discussed that the β-distribution reduct's problem of loss of hidden classification information is generated because that, in the procedure of calculating the β-distribution reduct, the differences among various conflicts are .

For a DT with $N = 1$, i.e., there is only one undecidable equivalent class (defined in Section IV), that means there is only one decision conflict in the decision rules of the original DT. Setting this undecidable equivalent class as C_i, according to eq.(3), $\forall x \in C_i$, x can support $\bigwedge(c, f(C_i, c)) \rightarrow (d, \emptyset)$; moreover, as there is only one decision conflict in the original DT, therefore, we need not consider the differences among various conflicts. When calculating the reduct, we can consider $\emptyset$ as a special decision

Table 3.4. Sample DT 3

U	α_1	α_2	α_3	α_4	α_5	d
x_1	1	1	1	1	1	1
x_2	1	1	0	1	1	1
x_3	0	0	1	0	0	1
x_4	1	1	2	1	1	1
x_5	1	1	0	1	0	2
x_6	1	1	0	1	1	2
x_7	0	0	1	2	1	2
x_8	1	1	0	1	1	2
x_9	1	1	2	1	1	2

class $D_\emptyset$ and set that $\forall D_j$, $D_j \neq D_\emptyset$. The sample DT 3 in Table 3.4 is an example.

In this DT, based on the values of the attributes, we have decision classes:

$D_1 = \{x_1, x_2, x_3, x_4\}$,

$D_2 = \{x_5, x_6, x_7, x_8, x_9\}$.

Also, we can obtain:

$C_1 = \{x_1\}$,

$C_2 = \{x_2, x_6, x_8\}$,

$C_3 = \{x_3\}$,

$C_4 = \{x_4, x_9\}$,

$C_5 = \{x_5\}$,

$C_6 = \{x_7\}$.

Set $\beta = 0.6$. Observing the sample DT 3, we can see that $C_4 = \{x_4, x_9\}$ is the only undecidable equivalent class, so we denote $C_4 \subseteq POS_C^{0.6}(D_\emptyset)$. Then we can see $\{\alpha_3, \alpha_4\}$ is a β-distribution reduct for this DT 3. In the original DT, the decision rules are:

$(\alpha_1, 1) \wedge (\alpha_2, 1) \wedge (\alpha_3, 1) \wedge (\alpha_4, 1) \wedge (\alpha_5, 1) \rightarrow (d, 1)$, supported by x_1;

$(\alpha_1, 1) \wedge (\alpha_2, 1) \wedge (\alpha_3, 0) \wedge (\alpha_4, 1) \wedge (\alpha_5, 1) \rightarrow (d, 2)$, supported by x_2, x_6 and x_8;

$(\alpha_1, 0) \wedge (\alpha_2, 0) \wedge (\alpha_3, 1) \wedge (\alpha_4, 0) \wedge (\alpha_5, 0) \rightarrow (d, 1)$, supported by x_3;

$(\alpha_1, 1) \wedge (\alpha_2, 1) \wedge (\alpha_3, 0) \wedge (\alpha_4, 1) \wedge (\alpha_5, 0) \rightarrow (d, 2)$, supported by x_5;

$(\alpha_1, 0) \wedge (\alpha_2, 0) \wedge (\alpha_3, 1) \wedge (\alpha_4, 2) \wedge (\alpha_5, 1) \rightarrow (d, 2)$, supported by x_7;

$(\alpha_1, 1) \wedge (\alpha_2, 1) \wedge (\alpha_3, 2) \wedge (\alpha_4, 1) \wedge (\alpha_5, 1) \rightarrow (d, (1or2))$, supported by x_4 and x_9.

After the reduction, the decision rules are:

$(\alpha_3, 1) \wedge (\alpha_4, 1) \rightarrow (d, 1)$, supported by x_1;

$(\alpha_3, 0) \wedge (\alpha_4, 1) \rightarrow (d, 2)$, supported by x_2, x_5, x_6 and x_8;

$(\alpha_3, 1) \wedge (\alpha_4, 0) \rightarrow (d, 1)$, supported by x_3;

$(\alpha_3, 1) \wedge (\alpha_4, 2) \rightarrow (d, 2)$, supported by x_7;

$(\alpha_3, 2) \wedge (\alpha_4, 1) \rightarrow (d, (1or2))$, supported by x_4 and x_9.

We can see that all the decision rules are kept consistent in the β-distribution reduct of sample DT 3 and no necessary information is omitted.

If a DT is β-inconsistent, $\beta \in (0.5, 1]$, use N to denote the number of undecidable equivalent classes, if $N > 1$, , the β-distribution reduct can keep the decision rule consistent but may lead to loss of hidden information in the original DT in some conditions.

If a DT is β-inconsistent, and there are more than one undecidable equivalent classes, the β-distribution reduct can avoid the decision rule conflict by keeping the

β-positive regions, or in another words, the decision part of the decision rules are unchanged. But in the meanwhile, the β-distribution reduct deals with all the undecidable equivalent classes by excluding them from all the β-positive regions, this may result that the β-distribution reduct will neglect the hidden differences among different undecidable equivalent classes, so that the β-distribution reduct may lose some necessary information or knowledge in DT and weaken the classification ability of the original system.

The sample DT in Table 3.3 discussed in Part B of Section III is an example to demonstrate this reduct problem in β-inconsistent table that has more than one undecidable equivalent classes.

3.5 The DT splitting method for β-complete reduct calculation

We have already discussed the relationship between the reduct and the β-consistent property of a DT in VPRS model. We see that in a VRPS model, set $\beta \in (0.5, 1]$, if a DT is 1-consistent, we only need to calculate the β-reduct of the DT; for a β-consistent DT, we have to calculate the β-distribution reduct to make sure the reduct not bringing in the decision rule conflicts; if a DT is β-inconsistent, the condition becomes complicated: if there are only one undecidable equivalent class, we only need to calculate the β-distribution reduct; Otherwise, only calculating the β-distribution reduct may lead to loss of some important information and lower the classification ability of the original DT. Accordingly, in this section, a new method will be proposed to deal with the reduct problem for the β-inconsistent DT with more than one undecidable equivalent classes.

In this section, we will first give a DT splitting method to deal with the reduct problem for β-inconsistent DT.

Theorem 3: In VPRS model, given $\beta \in (0.5, 1]$, if a DT is β-inconsistent, it can be split in two DTs, one is a β-consistent DT, the other one is a complete β-inconsistent DT in which all equivalent classes are undecidable.

Proof: If a DT $S = (U, A = C \cup D, V, f)$ is β-inconsistent, from eq.(21), $\exists C_i$, $\left|R^{\beta}(C_i)\right| = 0$. Since we have proved that $\left|R^{\beta}(C_i)\right| \in \{0, 1\}$ in Theorem 2, we can split the DT into two DTs S_1 and S_2. Elements in S_1 satisfy $\{x \,|x \in C_i, \left|R^{\beta}(C_i)\right| = 1\}$. Thus, $\forall C_i \in S_1, \left|R^{\beta}(C_i)\right| = 1$. According to eq.(21), S_1 is β-consistent. In the meanwhile, elements in S_2 satisfy $\{x \,|x \in C_i, \left|R^{\beta}(C_i)\right| = 0\}$, that means all equivalent classes are undecidable.

Based on Theorem 3, a β-inconsistent DT can be split according to the method shown in Figure 3.1.

Figure 3.2 uses sample DT 2 in Table 3.3 to demonstrate this DT splitting method with $\beta = 0.6$. We can see that sample DT 2 is split into a β-consistent DT, colored in white, and formed by the consistent equivalent classes; and a complete β-inconsistent DT, colored in dark, and formed by elements that are all in conflict.

Now we can provide the method for all the reduct (denoted as RED) of a β-inconsistent DT in the following steps:

Input: A DT: $S = (U, A = C \cup D, V, f)$, the precision parameter of the VPRS: β

Output: the reduct of the input DT RED

Step 1: Find all the equivalent classes C_i in this DT, where $\{C_i, i = 1, 2, \cdots, m\} = U/IND(C)$;

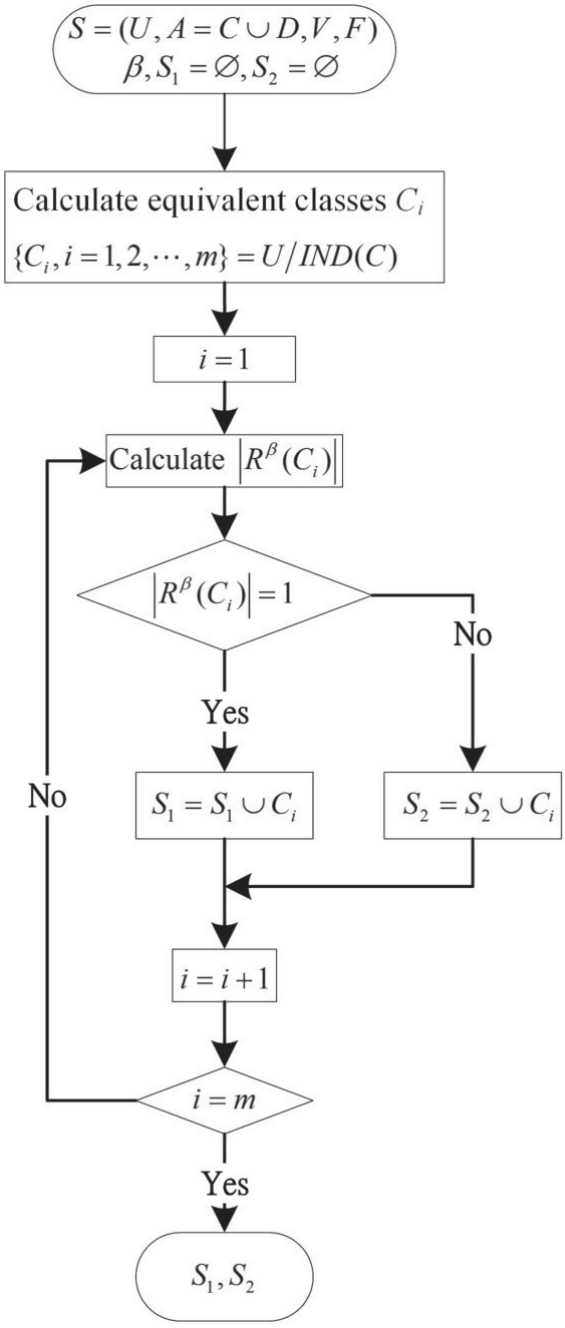


Figure 3.1. The β-inconsistent DT splitting approach

Step 2: For each C_i, calculate $\left|R^{\beta}(C_i)\right|$, and use N to denote the number of C_i, whose $\left|R^{\beta}(C_i)\right| = 0$;

Step 3: If $N = 1$, consider that the only undecidable equivalent class can be classified into $POS_c^{\beta}(D_{\emptyset})$, $D_{\emptyset} \neq D_j$, where $\{D_j, j = 1, 2, \cdots, n\} = U/IND(C)$. Then calculate the β-distribution reduct of this DT, a detailed algorithm for β-distribution reduct is given by [53], return the result as RED;

Step 4: If $N > 1$, split the DT in a β-consistent DT S_1 and a complete β-inconsistent DT S_2;

Step 5: Calculate a β-distribution reduct for S_1, and denote the reduct as RED_{S_1};

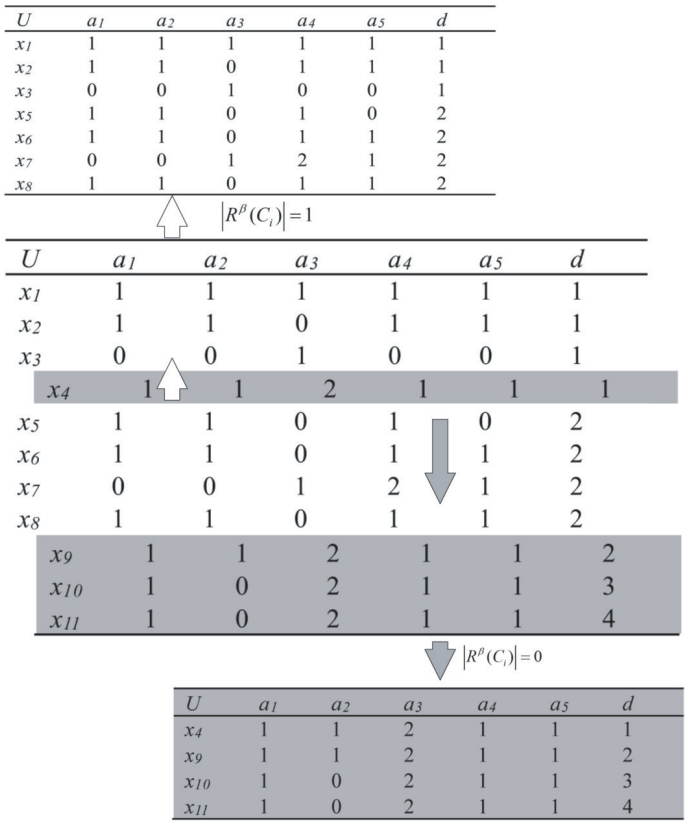
Step 6: For DT S_2, only consider the condition attributes, a DT without decision part is called Attribute-value Table (AT) [57], convert S_2 into an AT by deleting its decision attributes and calculate the reduct of this AT according to [51], denoted as RED_{S_2} ;

Step 7: return $RED = RED_{S_1} \cup RED_{S_2}$.

Using the sample DT 2 as an illustrative example, we split sample DT 2 in Table 3.3 into two DTs as shown in Figure 3.2. For the β-consistent DT which is colored in white in Figure 2, we can obtain its reduct $RED_{S_1} = \{\alpha_3, \alpha_4\}$; in the meanwhile, the reduct of the complete β-inconsistent DT colored in dark is $RED_{S_2} = \{\alpha_2\}$, accordingly, $RED = RED_{S_1} \cup RED_{S_2} = \{\alpha_2, \alpha_3, \alpha_4\}$, Table 3.5 shows the final obtained reduct with our proposed method.

From the original sample DT 2 in Part B, Section III, with $\beta = 0.6$, the total decision rules obtained are:

$(\alpha_1, 1) \wedge (\alpha_2, 1) \wedge (\alpha_3, 1) \wedge (\alpha_4, 1) \wedge (\alpha_5, 1) \rightarrow (d, 1)$, supported by x_1;



U	a_1	a_2	a_3	a_4	a_5	d
x_1	1	1	1	1	1	1
x_2	1	1	0	1	1	1
x_3	0	0	1	0	0	1
x_5	1	1	0	1	0	2
x_6	1	1	0	1	1	2
x_7	0	0	1	2	1	2
x_8	1	1	0	1	1	2

U	a_1	a_2	a_3	a_4	a_5	d
x_1	1	1	1	1	1	1
x_2	1	1	0	1	1	1
x_3	0	0	1	0	0	1
x_4	1	1	2	1	1	1
x_5	1	1	0	1	0	2
x_6	1	1	0	1	1	2
x_7	0	0	1	2	1	2
x_8	1	1	0	1	1	2
x_9	1	1	2	1	1	2
x_{10}	1	0	2	1	1	3
x_{11}	1	0	2	1	1	4

U	a_1	a_2	a_3	a_4	a_5	d
x_4	1	1	2	1	1	1
x_9	1	1	2	1	1	2
x_{10}	1	0	2	1	1	3
x_{11}	1	0	2	1	1	4

Figure 3.2. The β-inconsistent DT splitting approach on sample DT 2

$(\alpha_1, 1) \wedge (\alpha_2, 1) \wedge (\alpha_3, 0) \wedge (\alpha_4, 1) \wedge (\alpha_5, 1) \rightarrow (d, 2)$, supported by x_2, x_6 and x_8;

$(\alpha_1, 0) \wedge (\alpha_2, 0) \wedge (\alpha_3, 1) \wedge (\alpha_4, 0) \wedge (\alpha_5, 0) \rightarrow (d, 1)$, supported by x_3;

$(\alpha_1, 1) \wedge (\alpha_2, 1) \wedge (\alpha_3, 0) \wedge (\alpha_4, 1) \wedge (\alpha_5, 0) \rightarrow (d, 2)$, supported by x_5;

$(\alpha_1, 0) \wedge (\alpha_2, 0) \wedge (\alpha_3, 1) \wedge (\alpha_4, 2) \wedge (\alpha_5, 1) \rightarrow (d, 2)$, supported by x_7;

$(\alpha_1, 1) \wedge (\alpha_2, 1) \wedge (\alpha_3, 2) \wedge (\alpha_4, 1) \wedge (\alpha_5, 1) \rightarrow (d, (1or2))$, supported by x_4 and x_9;

$(\alpha_1, 1) \wedge (\alpha_2, 0) \wedge (\alpha_3, 2) \wedge (\alpha_4, 1) \wedge (\alpha_5, 1) \rightarrow (d, (3or4))$, supported by $x_1 0$ and $x_1 1$.

Table 3.5. The β-complete reduct DT of sample DT 2

U	α_2	α_3	α_4	d
x_1	1	1	1	1
x_2	1	0	1	1
x_3	0	1	0	1
x_4	1	2	1	1
x_5	1	0	1	2
x_6	1	0	1	2
x_7	0	1	2	2
x_8	1	0	1	2
x_9	1	2	1	2
x_{10}	0	2	1	3
x_{11}	0	2	1	4

and with $\beta = 0.6$, the total decision rules extracted from the reduct DT in table 3.5 are:

$(\alpha_2, 1) \wedge (\alpha_3, 1) \wedge (\alpha_4, 1) \rightarrow (d, 1)$, supported by x_1;

$(\alpha_2, 0) \wedge (\alpha_3, 1) \wedge (\alpha_4, 0) \rightarrow (d, 1)$, supported by x_3;

$(\alpha_2, 1) \wedge (\alpha_3, 0) \wedge (\alpha_4, 1) \rightarrow (d, 2)$, supported by x_2, x_6 ,x_5 and x_8;

$(\alpha_2, 0) \wedge (\alpha_3, 1) \wedge (\alpha_4, 2) \rightarrow (d, 2)$, supported by x_7;

$(\alpha_2, 1) \wedge (\alpha_3, 2) \wedge (\alpha_4, 1) \rightarrow (d, (1or2))$, supported by x_4 and x_9;

$\wedge(\alpha_2, 0) \wedge (\alpha_3, 2) \wedge (\alpha_4, 1) \rightarrow (d, (3or4))$, supported by x_10 and x_11.

Comparing the decision rules extracted from the original DT with those from the obtained reduct, we can see that the obtained reduct $RED = \{\alpha_2, \alpha_3, \alpha_4\}$ can not only keep the decision rule consistent, but also discern the two different undecidable equivalent classes $C_4 = \{x_4, x_9\}$ and $C_7 = \{x_{10}, x_{11}\}$ and conserve the hidden information that, given the condition $(, 2) \wedge (\alpha_4, 1)$, a sample point x with $f(x, \alpha_2) = 1$ has 50% prob-

ability to be classified in D_1 and 50% probability to be classified in D_2, and a sample point with $f(x, \alpha_2) = 0$ has 50% probability to be classified in D_3 and 50% probability to be classified in D_4.

The obtained reduct by this approach is called as the β-complete reduct in our paper. DT S_1 contains all the consistent equivalent classes in the original DT, accordingly, the β-distribution reduct of S_1 can keep all decision rules in original DT consistent. In the meanwhile, reduct of S_2 reflects the hidden classification information of the original DT. Then we get the union set of the two reducts as the β-complete reduct. It not only makes sure the β-positive regions are consistent in the reduct, but also considers the hidden information in the inconsistent equivalent classes of the original DT in order to avoid the deterioration of the original DT's classification ability.

Obviously, the requirement of the β-complete reduct is more rigorous than that of the β-distribution reduct. We may observe that the β-distribution reduct is a subset of the β-complete reduct, that means a β-complete reduct must be a β-distribution reduct, but the inverse proposition is not aways true.

3.6 An application in the real word

The table in Figure 3.3 was collected from the China Statistical Yearbook, 2009, published by the National Bureau of Statistics of China. The table contains the historical data of electricity power output and its influencing factors during 1978 to 2008 in China. The columns from the left to the right correspond to:

year; d :electricity power output (a hundred million kilowatt-hours), which will be considered as the decision attribute; α_1:GDP(a hundred million Yuans), α_2:Gross prod-

uct in primary industry(a hundred million Yuans), α_3: Gross product in second industry(a hundred millions Yuans), α_4: Gross product in third industry(a hundred millions Yuans), α_5: Per capita GDP(Yuans), α_6: Crude oil yield(ten thousand tons), α_7:Pig iron yield(ten thousand tons), α_8: Raw coal yield(a hundred million tons), α_9: traffic volume of railway(ten thousand tons); $\alpha_1 \quad \beta$ are condition attributes which are factors likely affecting the electronic power output. We try to use the VPRS theory to get a reduct of the decision table to delete the redundant factors which are less important to the China electronic power output and find out factors which are more significant to the electricity power output in China.

In the DT in Figure 3.3, the decision attributes are continuous variables. Accordingly, before we calculate the reduct of this decision table, we have to do discretization to the values of the decision table. The method we used is discretization of continuous attributes based on dynamic layer cluster . We first use systemic cluster method to cluster each attribute of this decision table into 3 4 classes, and then give each class a label and try to make sure the decision table is consistent. Figure 3.4 shows the decision table after the discretization process.

The choice of the value ofthe precision threshold β is another problem that needs to be solved. Some research has been done to try to find the best value of β, , but it is still lacking of a systematical method to fix the value. In most real applications, $\beta$is often given as an empirical value. Since the relationship between the factors in our case is quite complex, more uncertainties need to be considered and more error is allowed, we set the value of β as 0.4, which is near to the upper limit of its allowed range.

Then we can calculate the reduct of this DT in VPRS model under the precision of 0.4 with the proposed method in the paper. Figure 3.5 shows the relation matrix of

year	d	a1	a2	a3	a4	a5	a6	a7	a8	a9
1978	2566	3645.2	1027.5	1745.2	872.5	381	10405	3479	6.18	110119
1979	2820	4062.6	1270.2	1913.5	878.9	419	10615	3673	6.35	111893
1980	3006	4545.6	1371.6	2192	982	463	10595	3802	6.2	111279
1981	3093	4889.5	1559.5	2255.5	1076.6	492	10122	3417	6.22	107673
1982	3277	5330.5	1777.4	2383	1163	528	10212	3551	6.66	113465
1983	3514	5985.6	1978.4	2646.2	1338.1	583	10607	3738	7.15	118784
1984	3770	7243.8	2316.1	3105.7	1786.3	695	11461	4001	7.89	124074
1985	4107	9040.7	2564.4	3866.6	2585	858	12490	4384	8.72	130709
1986	4495	10274.4	2788.7	4492.7	2993.8	963	13069	5064	8.94	135635
1987	4973	12050.6	3233	5251.6	3575	1112	13414	5503	9.28	140653
1988	5452	15036.8	3865.4	6587.2	4590.3	1366	13705	5704	9.8	144948
1989	5848	17000.9	4265.9	7278	5448.4	1519	13764	5820	10.54	151489
1990	6212	18718.3	5062	7717.4	5888.4	1644	13831	6238	10.8	150681
1991	6775	21826.2	5342.2	9102.2	7337.1	1893	14099	6785	10.87	152893
1992	7539	26937.3	5866.6	11699.6	9357.4	2311	14210	7589	11.16	157627
1993	8395	35260	6963.8	16454.4	11915.7	2998	14524	8739	11.5	162794
1994	9281	48108.5	9572.7	22445.4	16179.8	4044	14608	9741	12.4	163216
1995	10070	59810.5	12135.8	28679.5	19978.5	5046	15004.95	10529.27	13.61	165982
1996	10813	70142	14015.4	33835	23326.2	5846	15733.39	10722.5	13.97	171024
1997	11356	78060.8	14441.9	37543	26988.1	6420	16074.14	11511.41	13.73	172149
1998	11670	83024.3	14817.6	39004.2	30580.5	6796	16100	11863.67	12.5	164309
1999	12393	88479.2	14770	41033.6	33873.4	7159	16000	12539.24	12.8	167554
2000	13556	98000.5	14944.7	45555.9	38714	7858	16300	13101.48	12.99	176581
2001	14808	108068.2	15781.3	49512.3	44361.6	8622	16395.87	15554.25	13.81	193189
2002	16540	119095.7	16537	53896.8	49898.9	9398	16700	17804.6	14.55	204956
2003	19105.75	134174	17381.7	62436.3	56004.7	10542	16959.98	21366.68	17.22	224248
2004	22033.09	159586.7	21412.7	73904.3	64591.3	12336	17587.33	26830.99	19.92	249296
2005	25002.6	184088.6	22420	87364.6	73432.9	14053	18135.29	34375.19	22.05	269296
2006	28657.26	213131.7	24040	103162	84721.4	16165	18476.57	41245.19	23.73	288224
2007	32815.53	250258.9	28627	124799	103879.6	19524	18631.82	47651.63	25.26	314237
2008	34668.82	302853.4	34000	146183.4	120486.6	22698	19001.24	47087.41	27.88	330354

Figure 3.3. The historical data of China electricity power output and its influencing factors from 1978 to 2008h

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	d
x_1	1	1	1	1	1	1	1	1	1	1
x_2	1	1	1	1	1	1	1	1	1	1
x_3	1	1	1	1	1	1	1	1	1	1
x_4	1	1	1	1	1	1	1	1	1	1
x_5	1	1	1	1	1	1	1	1	1	1
x_6	1	1	1	1	1	1	1	1	1	1
x_7	1	1	1	1	1	1	1	1	1	1
x_8	1	1	1	1	1	1	1	2	1	1
x_9	1	1	1	1	1	2	1	2	1	1
x_{10}	1	1	1	1	1	2	1	2	1	1
x_{11}	1	1	1	1	1	2	1	2	1	1
x_{12}	1	1	1	1	1	2	1	2	2	1
x_{13}	1	1	1	1	1	2	1	2	2	1
x_{14}	1	1	1	1	1	2	1	2	2	1
x_{15}	1	1	1	1	1	2	1	2	2	2
x_{16}	1	1	1	1	1	2	1	2	2	2
x_{17}	2	2	1	1	1	2	1	2	2	2
x_{18}	2	2	2	1	2	2	1	2	2	2
x_{19}	2	2	2	1	2	3	1	2	2	2
x_{20}	2	2	2	2	2	3	1	2	2	2
x_{21}	2	2	2	2	2	3	1	2	2	2
x_{22}	2	2	2	2	2	3	1	2	2	2
x_{23}	2	2	2	2	2	3	1	2	2	2
x_{24}	2	2	2	2	2	3	2	2	2	2
x_{25}	3	2	2	2	2	3	2	2	2	2
x_{26}	3	2	2	2	2	3	2	3	3	3
x_{27}	3	2	2	2	3	4	3	3	3	3
x_{28}	3	2	3	3	3	4	3	3	3	3
x_{29}	4	2	3	3	3	4	4	4	3	3
x_{30}	4	3	4	4	4	4	4	4	4	3
x_{31}	4	3	4	4	4	4	4	4	4	3

Figure 3.4. The final decision table after the discretization process

the DT. We will find that the sum of the elements in a1 is the largest and then we can delete the rows whose elements in columns of a1 get the values of 1, then we can find the column of β_9 meets the requirements of complement strategy 1, So, the result is $RED = \beta_1, \beta 9$. That means the GDP and the traffic volumes of rail way are two factors which are most significant to the electricity power output of China.

we can also give well economic explanation to the reduct of the decision table : that the GDP and traffic volume of railway are the main affect factors of the electricity power output is accordant with the economic signification in China. One is that the GDP has a high correlation ship with the demand of the electricity power in China. The other one is that the main electricity power in China is thermoelectricity (more than 75%) which highly depends on the supplement of coal, however, in China, only very a few provinces can supply coal to other provinces for the production of thermoelectricity, thus, the delivery of coal becomes an important affecting factor for the electricity power output. The main coal delivery style in China is the rail way, so the traffic volume of railway is a main affect factor to the power output. For the deleted attributes, we can also give some economic explanation. The crude oil yield cannot be seen as the consumption of the crude oil since that China imports a large number of crude oil every year, the oil yield is only a small part of the consumption; similarly, the raw coal yield is not equal to the quaint of coal used for thermoelectricity. In some years, the raw coal yield declined but the power output still increased, so the raw coal yield is not representative as an affect factor in this case.

	a1	a2	a3	a4	a5	a6	a7	a8
c1, c5	1	1	0	0	0	1	0	1
c2, c5	1	1	0	0	0	1	0	0
c3, c5	1	1	0	0	0	0	0	0
c4, c5	1	1	0	0	0	0	0	0
c1, c6	1	1	1	0	1	1	0	1
c2, c6	1	1	1	0	1	1	0	0
c3, c6	1	1	1	0	1	0	0	0
c4, c6	1	1	1	0	1	0	0	0
c1, c7	1	1	1	0	1	1	0	1
c2, c7	1	1	1	0	1	1	0	0
c3, c7	1	1	1	0	1	1	0	0
c4, c7	1	1	1	0	1	1	0	0
c1, c8	1	1	1	1	1	1	0	1
c2, c8	1	1	1	1	1	1	0	0
c3, c8	1	1	1	1	1	1	0	0
c4, c8	1	1	1	1	1	1	0	0
c1, c9	1	1	1	1	1	1	1	1
c2, c9	1	1	1	1	1	1	1	0
c3, c9	1	1	1	1	1	1	1	0
c4, c9	1	1	1	1	1	1	1	0
c1, c10	1	1	1	1	1	1	1	1
c2, c10	1	1	1	1	1	1	1	0
c3, c10	1	1	1	1	1	1	1	0
c4, c10	1	1	1	1	1	1	1	0
c1, c11	1	1	1	1	1	1	1	1
c2, c11	1	1	1	1	1	1	1	1
c3, c11	1	1	1	1	1	1	1	1
c4, c11	1	1	1	1	1	1	1	1
c1, c12	1	1	1	1	1	1	1	1
c2, c12	1	1	1	1	1	1	1	1
c3, c12	1	1	1	1	1	1	1	1
c4, c12	1	1	1	1	1	1	1	1
c1, c13	1	1	1	1	1	1	1	1
c2, c13	1	1	1	1	1	1	1	1
c3, c13	1	1	1	1	1	1	1	1
c4, c13	1	1	1	1	1	1	1	1
c1, c14	1	1	1	1	1	1	1	1
c2, c14	1	1	1	1	1	1	1	1
c3, c14	1	1	1	1	1	1	1	1
c4, c14	1	1	1	1	1	1	1	1
c1, c15	1	1	1	1	1	1	1	1
c2, c15	1	1	1	1	1	1	1	1
c3, c15	1	1	1	1	1	1	1	1
c4, c15	1	1	1	1	1	1	1	1
c5. c11	1	0	1	1	1	1	1	1
c6, c11	1	0	0	1	0	1	1	1
c7, c11	1	0	0	1	0	0	1	1
c8. c11	1	0	0	0	0	0	1	1
c9. c11	1	0	0	0	0	0	0	1
c10. c11	0	0	0	0	0	0	0	1
c5. c12	1	0	1	1	1	1	1	1
c6, c12	1	0	0	1	1	1	1	1
c7, c12	1	0	0	1	1	1	1	1
c8. c12	1	0	0	0	1	1	1	1
c9. c12	1	0	0	0	1	1	1	1
c10. c12	0	0	0	0	1	1	1	1
c5. c13	1	0	1	1	1	1	1	1
c6, c13	1	0	1	1	1	1	1	1
c7, c13	1	0	1	1	1	1	1	1
c8. c13	1	0	1	1	1	1	1	1
c9. c13	1	0	1	1	1	1	1	1
c10. c13	0	0	1	1	1	1	1	1
c5. c14	1	0	1	1	1	1	1	1
c6, c14	1	0	1	1	1	1	1	1
c7, c14	1	0	1	1	1	1	1	1
c8, c14	1	0	1	1	1	1	1	1
c9, c14	1	0	1	1	1	1	1	1
c10, c14	1	0	1	1	1	1	1	1
c5, c15	1	1	1	1	1	1	1	1
c6, c15	1	1	1	1	1	1	1	1
c7, c15	1	1	1	1	1	1	1	1
c8, c15	1	1	1	1	1	1	1	1
c9, c15	1	1	1	1	1	1	1	1
c10, c15	1	1	1	1	1	1	1	1

Figure 3.5. The relation matrix of the decision table

3.7 Conclusion

The β-distribution reduct is seen as a modified version of the β-reduct in the VPRS model. However, in our investigation, from some instances, we find that the β-distribution reduct also has limitations. It may neglect the differences among the different conflicts of a DT. Accordingly, a β-complement reduct is proposed in this chapter. By splitting the β-inconsistent DT into two DTs and combining the reduct of the two DTs together, we can obtain the β-complement reduct, and this β-complement reduct can avoid the loss of hidden classification information of the original DT.

We also analyze the hierarchical relationship between the proposed β-consistent notion and different definitions of reduct in VPRS model. For 1-consistent DT we only needs to calculate the β-reduct, if $\beta \in (0.5, 1)$, the β-distribution reduct can perform well for a β-consistent DT. For β-inconsistent DT, if there is only one undecidable class, we need to calculate the β-distribution reduct; otherwise, we need to split the β-inconsistent DT to get the β-complete reduct. Based on this investigation, for a given DT and VPRS with certain β, we can analysis the β-consistency property of this DT first, and then according to the β-consistency property, we choose the proper definition of reduct for this DT, and calculate it with suitable algorithm in [50, 51] or [53].

Chapter 4

Efficient Feature Learning for RBFNN

The feature representation by NNs is becoming the most popular research topics in the past ten years. Lastest studies are main focuse on using CNN family of models to learn the features. However, this kind of approach are more suitable for image processing or language processing application. Other applications, such as some data mining tasks, are still waiting for suitable NN tools to solve the feature representation problem.

The Radial Basis Function Neural Network (RBFNN) model [58, 59] is an important research branch of Feedforward Neural Network (FNN). The Radial Basis Function (RBF) was traditionally used as a method for function interpolation tasks in multidimensional space [60]. However, for the scenario where thousands of noisy data points are involved, an approximative solution to the data is more desirable than an interpolative one. By reducing the number of basis functions and giving a more suitable basis, Broomhead and Lowe proposed a more general RBF approach for function approxi-

mation. A more general architecture was consequently proposed as a Neural Network model for classification tasks [59,61]. A typical RBFNN consists of three layers: an input layer, a hidden layer with Radial Basis Functions (RBF) based nonlinear mappings, and an output layer. Under some additional conditions imposed on the basis functions, the set of RBFNN with freely adjustable prototype vectors are shown to be universal approximators, so that any continuous function can be approximated with arbitrary precision [61]. Therefore, the RBFNN theoretically can offer approximation capabilities similar to other types of FNNs.

For a function interpolation task, the RBF centers should be the known points in the given data set. In an RBFNN architecture, the number of RBF neurons should be less than the number of the given data samples in order to reduce the complexity of the model to avoid the overfitting. Moreover, the centers of the RBFs are not restricted to be the data points but usually calculated via algorithms in the training process. In the network architecture, the RBF centers are considered as representative prototypes of the data samples in the feature space. The output of a certain input sample is determined by the relationship among this sample and different prototypes in the hidden layer. Accordingly, the performance of an RBFNN critically depends upon the chosen centers [62].

Since the discriminative power is determined by RBF centers, the calculation of the RBF centers plays an important role in the entire RBFNN training scheme. Theoretically the centers can be calculated together with the network weights via backpropagation [63]. In practice, such training scheme is seldom employed. The converging of the network may become more slowly and apparently towards ultimately poorer local minima when this training scheme is adopted. In most cases, RBFNN models are trained via a two-phase learning scheme [61]. In the first phase, the center vectors of

the RBF in the hidden layer are determined. The network weights can be calculated by fitting a linear model of the hidden layer's outputs with respect to the adopted objective function in the second phase. In some applications, a back-propagation step is suggested as the third phase to fine-tune all of the parameters in an RBFNN [64].

Generally, the RBF centers can be calculated via either supervised or unsupervised methods. For the supervised aspect, the RBF centers and weights can be estimated according to the given objective function simultaneously by back-propagation. Other supervised methods include the Orthogonal Least Squares (OLS) learning algorithm, the Support Vector (SV) approach, and the Learning Vector Quantization (LVQ) [65, 66]. In a more general case, the unsupervised learning approach is adopted to determine the RBF centers. Clustering algorithm, especially the k-means, has taken the dominant position in unsupervised approaches for RBF center training. The objective of using the clustering approach is actually to determine the representative prototypes for the input data set in order to minimize the classical quantization error [67]. The advantages of identifying the RBF centers via clustering approaches include guiding the learning towards a local minima that supports better generalization from the training data and avoid the overfitting [68]. Another branch of unsupervised learning approach is to obtain the RBF centers via some transformation of the given data set, e.g., assigning the Karhunen-Loeve transform (KLT) scaled eigenvectors to the RBFNN centers to lower prediction normalized mean squared error [69].

An RBFNN with k-means determined RBF centers usually offers a satisfactory accuracy in real applications. However, one problem of it is that the clustering algorithm is usually very time-consuming. Meanwhile, other algorithms that can reduce the training time of determining the RBF centers may lower the output accuracy of the entire RBFNN model. For example, some studies directly employ randomly generated vectors

as the RBF centers for greatly reducing the training time [70, 71]. The randomly generated center vectors, however, cannot reliably determine the representative prototypes in the input feature spaces. As a result, the performance of the entire RBFNN is also adversely affected.

Concentrating on reducing the training time without a significant loss of the reliability, this chapter proposes a new approach for determining the RBF centers for an RBFNN. In a two-phase training scheme, an eigenvector-based clustering analysis technique is implemented first to determine the RBF centers in the feature space of the training samples. The network weights are then calculated either by pseudo-inverse solutions or gradient descent algorithms. By considering the principal components of the data set as the relax solution of k-means, the proposed approach determines the RBF centers much faster than the mainstream clustering algorithms. Moreover, it can keep the clustering error within a bound to attain a comparable accuracy. In advance, compared with supervised approaches, the proposed approach enables the RBFNNs to offer higher generalization in the experiments by avoiding the poorer local minima. In general, the main contribution of this chapter is providing a more efficient method of calculating the RBF centers for training up an RBFNN model. The efficiency and reliability of the proposed approach are demonstrated in the experimental results.

The remaining part of this chapter is organized as follows. Section 4.1 introduces some background knowledge about the network architecture of RBFNN, including a discussion of the current training approaches. Related works of determining the RBF centers are also given in this section. Section 4.2 presents the idea of the proposed approach, which uses the eigenvector-based method to determine the RBF centers in the feature space of the training samples and then calculates the output weights of the model. Section 4.3 describes the experiments and shows the comparison results. The

Table 4.1. Basic mathematical notations

notation	Definition
d	The number of input variables of an RBFNN
D	The number of RBF neurons of an RBFNN
L	The number of output neurons of an RBFNN
k	The indices of the RBF neurons of an RBFNN, $k = 1, \ldots, D$
j	The indices of the output neurons of an RBFNN, $j = 1, \ldots, L$
X	A training sample with d dimensions
c_k	The RBF center for the kth RBF neuron and the cluster centroids of k-means
r_k	The width parameter for the kth RBF neuron
w_{kj}	The network connecting weight between the kth RBF neuron and the jth output neuron
K	The number of clusters for k-means (In an RBFNN training scheme, K is set as D)
C_k	The kth cluster via k-means
n	The number of samples
$\mathbf{X}$	The data matrix with n d-dimensional samples, $\mathbf{X} = (X_1, X_2, ..., X_n)^{\mathrm{T}}$, $X_i \in \mathbb{R}^d$, $i = 1, 2, \ldots, n$
i	The indices of the samples, $i = 1, \ldots, n$
$\bar{X}$	The vector of feature means of $\mathbf{X}$, $\bar{X} = \sum_i^n X_i/n$
$\tilde{\mathbf{X}}$	The centered data matrix by removing the means, $\tilde{\mathbf{X}} = (\tilde{X}_1, \tilde{X}_2, ..., \tilde{X}_n)^{\mathrm{T}}$, where $\tilde{X}_i = X_i - \bar{X}$
u, v	The eigenvector vectors of $\tilde{\mathbf{X}}^{\mathrm{T}}\tilde{\mathbf{X}}$ and$\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}}$
J_K	The objective function of k-means
$I(C_k)$	The membership indicator for the kth cluster

conclusion is given in the last section, together with the discussion of some potential future work.

4.1 Background Knowledge

In this section, the background knowledge utilized in the study is briefly introduced, including a brief review of the RBFNN. Moreover, the idea of two- and three-phase training approaches are also reviewed. For convenience, the basic mathematical notations are given in Table 4.1.

4.1.1 A Brief Introduction of the RBFNN

An RBFNN [60] is a single-hidden-layer FNN model. Fig.4.1 gives a three-layer architecture of the RBFNN model that consists of d inputs, D hidden neurons, and L

output units. In the hidden layer, the kth ($k = 1, 2, ..., D$) hidden neurons are associated with a center c_k that is equal in dimension with the number of input variables. A positive real number for scaling, r, which is called the scaling parameter or the width of the RBF, is also given to each hidden neuron. In each hidden neuron, by applying the RBF, which is a radial symmetric function usually according to the Euclidean norm of the difference between the data point $X \in \mathbb{R}^d$ and $c_k \in \mathbb{R}^d$, the input data is represented in a new feature space.

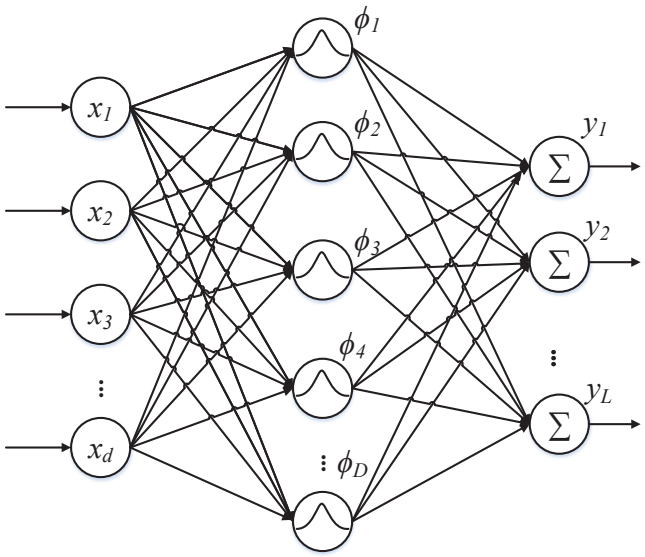


Figure 4.1. Typical structure of a RBFNN with d-dimensional input data, D hidden neurons, and L-dimensional output

The most commonly used RBF function in the kth RBF neuron is

$$\phi_k(X) = \exp(-\frac{\|X - c_k\|^2}{2r_k^2}). \tag{4.1}$$

The final output of a certain data point X in the jth ($j = 1, 2, \ldots, L$) output unit of the RBFNN is produced by the linear combination of the responses of hidden neurons as

$$\hat{y_j}(\phi(x)) = \sum_{k=1}^{D} w_{kj} \cdot \phi_k(X) = \sum_{k=1}^{D} w_{kj} \cdot \exp(-\frac{\|X - c_k\|^2}{2r_k^2}), \tag{4.2}$$

where w_{kj} denotes the weight connecting the kth hidden neuron and the jth output unit. The L-dimensional output $(\hat{y_1}, \ldots, \hat{y_L})$ is denoted as $\hat{Y}$. Most commonly, for n training samples, the network is optimized via minimizing the least squares loss function

$$J_R = \sum_{i=1}^{n} \|Y_i - \hat{Y}_i\|^2, \tag{4.3}$$

where Y_i is the real value for the ith sample and $\hat{Y}_i$ is the network output for the ith sample, $i = 1, \ldots, n$.

Similar to typical sigmoid NNs, RBFNNs can be trained via the back-propagation approaches according to the gradient of the objective functions. The adjusting rules of the RBF centers and the weights are

$$\Delta c_k = \eta \frac{w_k}{r^2} E \sum_{j=1}^{L} \phi_k(X)(X - c_k) \tag{4.4}$$

and

$$\Delta w_k = \eta E \sum_{j=1}^{L} \phi_k(X), \tag{4.5}$$

where E is the training error in each iteration, which is usually defined by the least squares objective function, and η is the learning rate.

4.1.2 The Training Schemes of RBFNN

Eq.4.4 and Eq.4.5 have shown that RBFNNs could be trained via back-propagation globally. In practice, directly training all the parameters simultaneously and iteratively may easily lead an RBFNN to a poor local minima. Therefore, in most of the cases, an RBFNN model is trained up with a two-phase or three-phase learning scheme.

In a two-phase learning scheme, an RBFNN can be trained via two stages [61]:

stage 1. Adjusting the parameters of the RBF neuron layer, mainly including the RBF centers $c_k \in \mathbb{R}^d, k = 1, 2, \ldots, D$, where d is the number of the input space dimensions and D is the number of the hidden neurons, i.e., the number of the dimensions of the mapped feature space.

stage 2. Calculating the output weights $w_k \in \mathbb{R}^L, k = 1, 2, \ldots, D$, of the model, where L is the number of output neurons.

In stage 1, to determine the centers for the RBFNN, typically unsupervised learning algorithms such as clustering algorithms are adopted. The RBF width parameters r_ks are usually all fixed to the same value which is proportional to the maximum distance between the chosen centers. In stage 2, the weights of the output layer can be calculated via supervised learning algorithms, e.g., gradient descent approach or pseudo-inverse solution.

After a two-phase learning is utilized for the initialization of an RBFNN model, in some applications, the whole RBFNN architecture can be optimized further. All the parameters are fine-tuned via the back-propagation according to Eq.4.4 and Eq.4.5. Such learning process is called the three-phase learning scheme.

4.1.3 Related Work for Determining the RBF Centers

According to the previous discussion, the output of an RBFNN is not only determined by the network weights but also the RBF centers in the hidden layer. Many studies have been conducted on efficiently and reliably determining the RBF centers.

The naive approach, which employs randomly generated RBF centers, has been reported to be able to provide an acceptable accuracy in early studies [70, 71]. Meanwhile, a globally supervised approach, which simultaneously estimates all the parameters (including the RBF centers and the connecting weights) of the entire network via back-propagation has also been adopted [63, 64]. However, both of these two training approaches are not prevalently employed. The former approach is not reliable enough while the latter one may lead the entire network towards ultimately poorer local minima.

In practice, the RBF centers can be obtained either with a supervised approach or unsupervised approach. For example, the OLS learning algorithm, which chooses the RBF centers one by one in a rational way until an adequate network has been constructed [72]. The SV approach is another supervised method for RBF centers selection which determines the structure of the classifier by minimizing the bounds of training error and generalization error [65]. Other supervised approaches for identifying RBF center include the family of Learning Vector Quantization (LVQ) algorithms which calculates the RBF centers according to the pre-defined class-memberships [66, 73], and the Fisher Ratio Class Separability Measure based RBF center selection method [62]. In practice, the SV approach is better known as Support Vector Machine (SVM) and not often considered as an RBFNN model. Meanwhile, the LVQ approach and the Fisher Ratio approach are also not widely employed. The unsupervised approach has taken the dominant place in finding the RBF centers.

The unsupervised learning approach of RBF center can be considered as a pre-training operation before estimating the connecting weights of the network. The advantages of such unsupervised pre-training, including the fast converging and avoidance from overfitting, have been discussed a lot in many important studies recently [68, 74]. For an RBFNN, when the unsupervised learning approach is adopted, clustering approaches are employed to determine the representative prototypes in the feature space. The feature space thus can be divided into many sub-regions represented by the RBF centers. Among many clusterings algorithms, the k-means [75] is the most widely used approach for determining the RBF centers via unsupervised learning [61, 76–78]. Another important unsupervised approach is to obtain the RBF centers via the KLT scaled eigenvectors of the data matrix [69], which adopts the relationship between the eigenvectors of the data matrix and the centers of the RBF neurons to improve the performance of the RBFNN.

4.1.4 Determining RBF Centers via k-means

The k-means clustering aims to partition all the data samples into k clusters. Thus each data point belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the feature space into Voronoi cells [79]. When using k-means to determine the RBF centers, for the D hidden neurons, the input feature space should be divided into D sub-regions. In each sub-region, the cluster centroid should be selected as an RBF center to represent the corresponding sub-region.

A typical k-means model locates the cluster centroids $c_k, k = 1, 2, \ldots, K$, for the K clusters $C_1, C_2, \ldots, C_K$ of a data set $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ with n samples by minimizing

$$J_K = \sum_{k=1}^{K} \sum_{X \in C_k} \|X - c_k\|^2. \tag{4.6}$$

For training the RBF centers in D neurons, the D corresponding clusters can be obtained via Eq.4.6 by setting $k = D$. Thus, the D clusters can divide the initial feature space into sub-regions. This can be considered as a priori knowledge for the following supervised learning phase of the network. However, the k-means approach is relatively time-consuming, the upper bound of the time complexity of k-means algorithm is $O(n \times d \times D \times t)$, where t is the number of iterations. When the number of samples, n, is large, this approach is rather time-consuming. The main objective of the method discussed in the following section is to provide a more efficient approach for determining the RBF centers. The centers should be obtained within a shorter time than k-means, while the obtained centers should represent sub-regions of the feature space for representative prototypes.

4.2 RBFNN Trained with The Eigenvector-based Fast RBF Center Selection

In this section, to provide a more efficient algorithm for obtaining the RBF centers, an approach to calculate the centers via the eigenvectors is introduced.

4.2.1 PCA for k-means

The proposed approach for determining the RBF centers is based on the relationship between Principal Component Analysis (PCA) and k-means clustering [80–82].

Here are some notations on PCA. Set $\mathbf{X} = (X_1, X_2, ..., X_n)^{\mathrm{T}}, X_i \in \mathbb{R}^d$, $i = 1, 2, \ldots, n$, represents the given data matrix with n data points in a d-dimensional feature space and $\tilde{\mathbf{X}} = (\tilde{X}_1, \tilde{X}_2, ..., \tilde{X}_n)^{\mathrm{T}}$, where $\tilde{X}_i = X_i - \bar{X}$ is the centered data matrix by removing the mean $\bar{X} = \sum_i^n X_i/n$ of the given data set. A principal direction u, which is the principal component of $\tilde{\mathbf{X}}^{\mathrm{T}}\tilde{\mathbf{X}}$, and a principal component v that are the eigenvectors satisfying

$$\tilde{\mathbf{X}}^{\mathrm{T}}\tilde{\mathbf{X}}u = \lambda u, \tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}}v = \lambda v. \tag{4.7}$$

The principal components (eigenvectors) can be considered as indicators of the cluster membership [83, 84]. The proof is given as the remaining part of this section.

First, when using the k-means to determine RBF centers, the obtained centers are expected to identify the subregions for the prototypes by minimizing the sum of the distance between each sample and the corresponding clustering centroids as in Eq.4.6. After some algebra, the loss function Eq.4.6 can be rewritten as

$$J_K = \sum_i^n {X_i}^2 - \sum_{k=1}^K \frac{1}{n_k} \sum_{X_i, X_{i'} \in C_k} X_i X_{i'}^{\mathrm{T}}, \tag{4.8}$$

where n_k is the number of samples in the k cluster [85].

According to [83], set non-negative vectors, $H_K = (h_1, h_2, \ldots, h_K)^{\mathrm{T}}$, where

$$h_k = (0, \ldots, 0, \overbrace{1, \ldots, 1}^{n_k}, 0, \ldots, 0)^{\mathrm{T}}/{n_k}^{1/2}, \tag{4.9}$$

the positive values in h_k indicate that the corresponding samples belong to the kth clus-

ter in the feature space. Putting Eq.4.8 and Eq.4.9 together, there is

$$J_K = \sum_{i}^{n} {X_i}^2 - \mathrm{Tr}(H_K \mathbf{X}\mathbf{X}^{\mathrm{T}} H_K^{\mathrm{T}}). \tag{4.10}$$

There are redundancies in H_K. Thus one of the h_ks is a linear combination of others. To remove the redundancy, a linear transformation T to H_K is given as

$$Q_K = H_K^{\mathrm{T}} T, \tag{4.11}$$

where T is a $K \times K$ orthogonal matrix with the last column being set as

$$t_K = (\sqrt{n_1/n}, \sqrt{n_2/n}, \ldots, \sqrt{n_K/n})^{\mathrm{T}}. \tag{4.12}$$

Thus the last column of Q_K will be

$$q_K = \sqrt{\frac{n_1}{n}} h_1 + \sqrt{\frac{n_2}{n}} h_2 + \cdots + \sqrt{\frac{n_K}{n}} h_K = \sqrt{\frac{1}{n}} e^{\mathrm{T}}. \tag{4.13}$$

From Eq.4.9, h_k has mutual orthogonality, i.e., $h_k h_l^{\mathrm{T}} = \delta_{kl}$, where $\delta_{kl} = 1$ if $k = l$, otherwise $\delta_{kl} = 0$. Thus $q_k^{\mathrm{T}} q_l = h_k (T^{\mathrm{T}} T) h_l^{\mathrm{T}} = \delta_{kl}$. Consequently, set $Q_{K-1} = (q_1, q_2, \ldots, q_{K-1})$, there is

$$Q_{K-1}^{\mathrm{T}} Q_{K-1} = I_{K-1} \tag{4.14}$$

and

$$q_k^{\mathrm{T}} e^{\mathrm{T}} = 0, \text{ for } k = 1, \ldots, K-1. \tag{4.15}$$

Eq.4.10 therefore can be transformed as

$$J_K = \sum_{i}^{n} {X_i}^2 - e\mathbf{X}\mathbf{X}^{\mathrm{T}}e^{\mathrm{T}}/n - \mathrm{Tr}({Q_{K-1}}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}Q_{K-1}). \tag{4.16}$$

Since $\tilde{\mathbf{X}} = \mathbf{X} - \bar{\mathbf{X}}$, according to [84], a further transformation of Eq.4.16 is given as

$$J_K = \mathrm{Tr}(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}}) - \mathrm{Tr}({Q_{K-1}}^{\mathrm{T}}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}}Q_{K-1}) = \mathrm{Tr}(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}}) - J_D. \tag{4.17}$$

Now minimizing J_K is equal to maximizing J_D, since the term $\mathrm{Tr}(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}})$ is always a constant. The solution for the optimization of J_D is given in the well known theorem by Ky Fan [86]:

THEOREM 4.1 (Fan) *Setting A is a symmetric matrix with eigenvalues $\varsigma_1 \geq \cdots \geq \varsigma_n$ and eigenvectors $(v_1, \ldots, v_n)$, the maximization of* $\mathrm{Tr}(Q^{\mathrm{T}}AQ)$*, which subject to constraints $Q^{\mathrm{T}}Q = I_K$ has the solution $Q = (v_1, \ldots, v_K)R$, where R is an arbitrary $K \times K$ orthonormal matrix, and* $\max \mathrm{Tr}(Q^{\mathrm{T}}AQ) = \varsigma_1 + \cdots + \varsigma_n$.

There are three constraints for the optimization of J_D, the Eq.4.14, Eq.4.15, and q_k are the linear transformations of the h_k as in Eq.4.11. By ignoring the last constraint, i.e., allow h_k to take continuous values, according to **Theorem 4.1**, the continuous solution of k-means clustering for the transformed discrete cluster membership indicator vectors Q_{K-1} is comprised of $K-1$ principal components: $Q_{K-1} = (v_1, \ldots, v_{K-1})$. And according to Eq.4.17, the error loss of the optimization has upper and lower bounds:

$$n\overline{\tilde{\mathbf{X}}^2} - \sum_{k=1}^{K} \lambda_k < J_K < n\overline{\tilde{\mathbf{X}}^2}, \tag{4.18}$$

where $n\overline{\tilde{\mathbf{X}}^2}$ is the total variance and λ_k are the principal eigenvalues of $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}}$.

Based on the previous discussion, PCA can provide a relax solution to get the membership indicators of the k-means clustering by employing the principal component. Here the term relax means the last constraint is loosen. According to [84], set the threshold as 0, for each cluster C_k, the eigenvectors $v_k = (v_{k1}, v_{k2}, \ldots, v_{kn})$ can indicate the memberships by

$$I(C_{k1}) = \{i | v_{ki} \leqslant 0\}, I(C_{k2}) = \{i | v_{ki} > 0\}, \tag{4.19}$$

where $I(C_{k1})$ and $I(C_{k2})$ are the set of indexes of data points that belong to the two sub-clusters C_{k1} and C_{k2} respectively of the whole data set regarding the kth cluster C_k. One of the two sub-clusters C_{k1} and C_{k2} represents the samples acceptable to the kth cluster, the other one represents the samples unacceptable to the kth cluster. Since we have the assumption that the data points for each class are equally distributed, the sub-cluster with a smaller size can represent the data points in kth cluster [87].

4.2.2 An Algorithm of the Proposed Method

Based on the previous discussion, the detailed algorithm of the proposed RBF center calculation approach is given in Algorithm 1.

Identifying the RBF centers is the first phase of an RBFNN scheme. In the second phase, the connecting weights should be calculated. In most scenarios, the loss function of the RBFNN is the least squares. Therefore, according to Eq.4.2, training up the network weights is equal to solving a linear matrix equation once the RBF centers are specified. Thus the weights can be analytically calculated via the pseudo-inverse since

Algorithm 4.1 An algorithm of determining the RBF centers via eigenvectors

Require: Given an n-sample training set $\mathcal{S} = \{(X_i, Y_i)|X_i \in R^d, Y_i \in R^L, i = 1, 2, \ldots n\}$, and the number of hidden neurons D;

Ensure: The RBF centers c_ks for the hidden nodes of the network.

1: Denote the data matrix as $\mathbf{X} = (X_1, X_2, \ldots, X_n)^{\mathrm{T}}$;
2: Shift the mean of the samples on each feature to zero by $X_i - \bar{X}$, where $\bar{X} = \sum_i^n X_i/n$, denote the results as $\tilde{\mathbf{X}} = (\tilde{X}_1, \tilde{X}_2, ..., \tilde{X}_n)^{\mathrm{T}}$;
3: Find the D eigenvectors $v_1, v_2, ...v_D$ of $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}}$, each eigenvector can be considered as a continuous solution of the membership indicator for the corresponding cluster.
4: Based on the obtained D eigenvectors, generate D membership indicator vectors according to Eq.4.19. For each indicator vector $I(C_k), k = 1, 2, \ldots, D$, find two sub-clusters C_{k1} and C_{k2} in $\mathbf{X}$, and use the smaller sub-cluster as the kth cluster C_k;
5: Calculate the mean of the data points found in C_k to get the RBF center c_k;
6: **return** c_k.

the solution is unique [59], or using the back-propagation hence the coefficients will converge to the optimal solution via gradient descent algorithm [64].

Here is some discussion on the proposed approach.

First of all, the time complexity of the k-means algorithm is $O(n \times d \times K \times t)$. For a large-scale data set with a large value of n, k-means will be very time-consuming. For the proposed approach, the time complexity depends on the specific PCA algorithm employed in the implementation. For PCA calculation, to obtain the eigenvectors, the time complexity will be $O(d^3)$. A typical machine learning problem usually requires $n > d$ [88]. Especially, for large-scaled data sets, there are $n \gg d$. Therefore, the proposed approach usually has less time complexity than k-means, and will be much more efficient than k-means for data sets relatively large in size.

Secondly, compared with k-means, since the proposed approach is based on the relax solution of the k-means loss function, the obtained centroids are less precise from the clustering point of view. According to Eq.4.18, the eigenvectors can be considered

as the relax solution of the k-means within a bounded error. However, from the whole network aspect, a relatively relax solution in the unsupervised training phase may be helpful. The less precise clustering results may help to avoid the poorer local minima so that the generalization of the model may be enhanced [89, 90]. The proposed approach is evaluated in the following experiments.

It means that we reduce the training time of the first training phase by sacrificing the training accuracy to some extent. However, recent investigations [89, 90] have demonstrated that the training accuracy of the first layer of feedforward neural networks maybe not as important as we thought traditionally. By adjusting the weights of the output layer, feedforword neural networks can provide comparable accuracy without fully training in the first layer. and the centers obtained via the proposed approach is enough to represent meaningful sub-regions of the feature space. Therefore, the proposed approach is reliable.

4.3 Experiments and Results

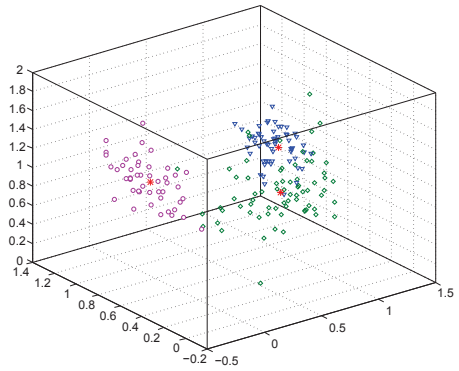
A series of experiments have been conducted to evaluate the performance of the proposed approach. The first experiment aims to visualize the RBF centers obtained via the traditional clustering approach and proposed approach in the feature space. The second experiment gives a comparison of the time consumption and the accuracy between three training schemes with different network scales. Finally the classification performance of differently trained RBFNNs are compared.

4.3.1 Clustering Results Visualization

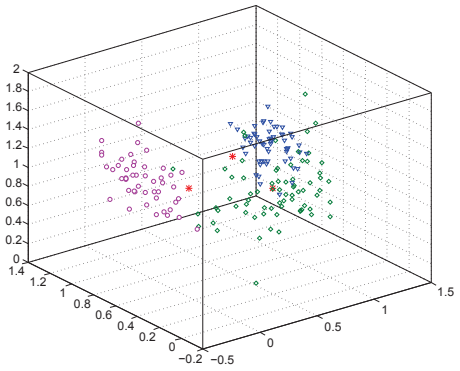
Five different benchmarking data sets collected from UC Irvine Machine Learning Repository (UCI) have been employed to conduct the first experiment [91]. For each data set, the typical k-means approach and the proposed eigenvector-based approach are adopted respectively for determining the RBF centers. The number of RBF centers is decided by the number of classes for each data set. This experiment is to visually examine whether the obtained centers can identify the partitions of the feature space for representative prototypes. The identified RBF centers can be employed in the subsequent supervised learning stage of an RBFNN. Fig.4.2 shows the comparison results of the two adopted approaches on five different data sets.

For each data set, the data points are drawn in a 3-D (three principal components obtained via PCA) feature space for visualization in each sub-figure of Fig.4.2. Data points with different class labels are distinguished via shapes and colors in each sub-figure. The obtained centroids are marked with big red stars for visualizing the results.

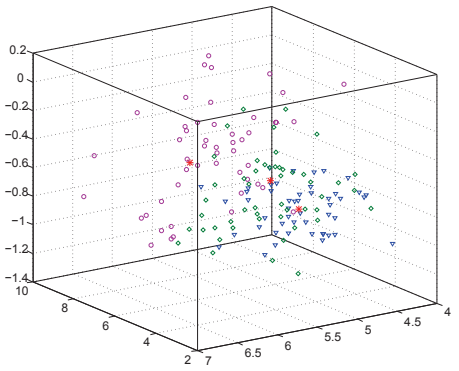
As shown in Fig.4.2, for the data sets of Auto, Inon and Parkinson, the centroids obtained via the eigenvector-base approach are very close to those obtained via k-means. For the other two data sets, the centroids obtained via the eigenvector-base approach are relatively far from those obtained via k-means. All the calculated centroids are scattered enough and each centroid can be located in a different cluster. i.e., the obtained centers can be reasonably considered as stereotypical patterns of the training samples. Specifically, in the experiment, for all the five data sets, the k-means algorithm spent at least 0.159 second on calculating the centers. Conversely, the proposed approach was able to calculate the centers within 0.01 second. A notable result is that for imbalanced data sets such as Parkinson, the centers obtained via the proposed approach are more



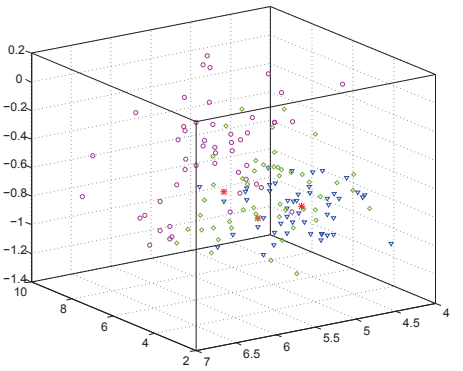
(a) Centers for Wine via k-means



(b) Centers for Wine via eigenvectors



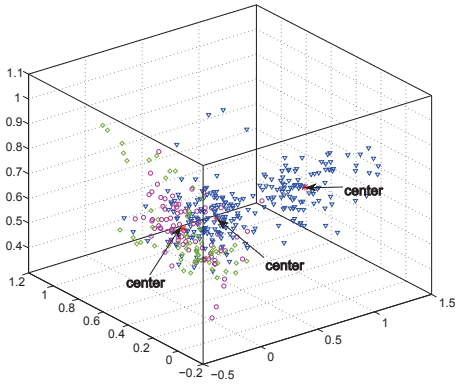
(c) Centers for Iris via k-means



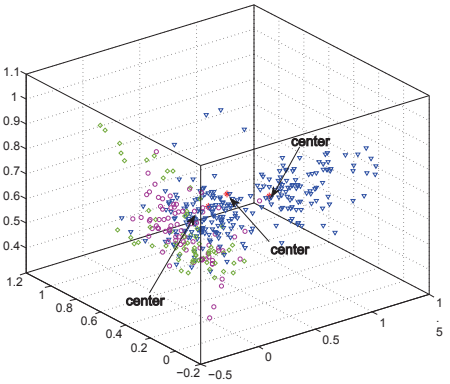
(d) Centers for Iris via eigenvectors

Figure 4.2. Centers Selection in Different Data Sets via k-means and Proposed Approach

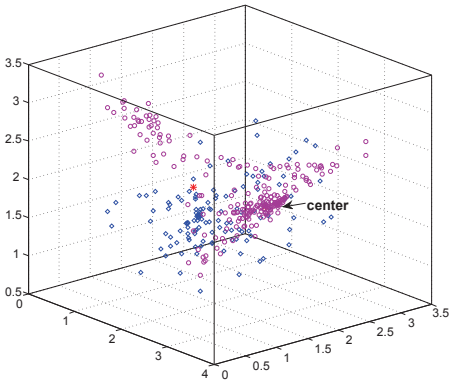
close to the real centers than that of the k-means approach.



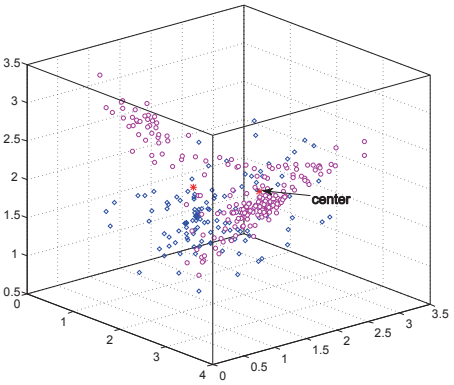
(e) Centers for Auto via k-means



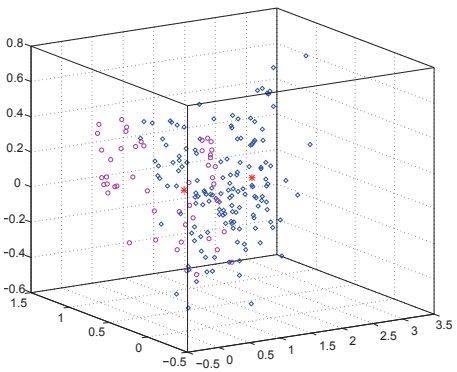
(f) Centers for Auto via eigenvectors



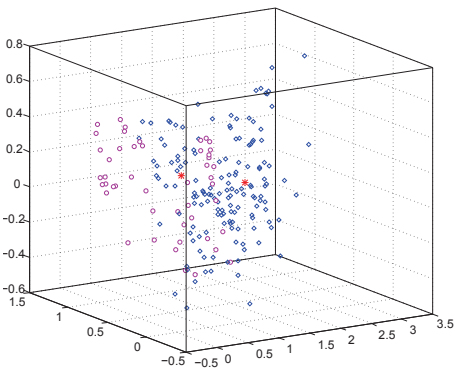
(g) Centers for Inon via k-means



(h) Centers for Inon via eigenvectors



(i) Centers for Parkinson via k-means



(j) Centers for Parkinson via eigenvectors

Figure 4.2. Centers Selection in Different Data Sets via k-means and Proposed Approach

4.3.2 Experimental Results on RBFNNs with Increasing Number of Hidden Neurons

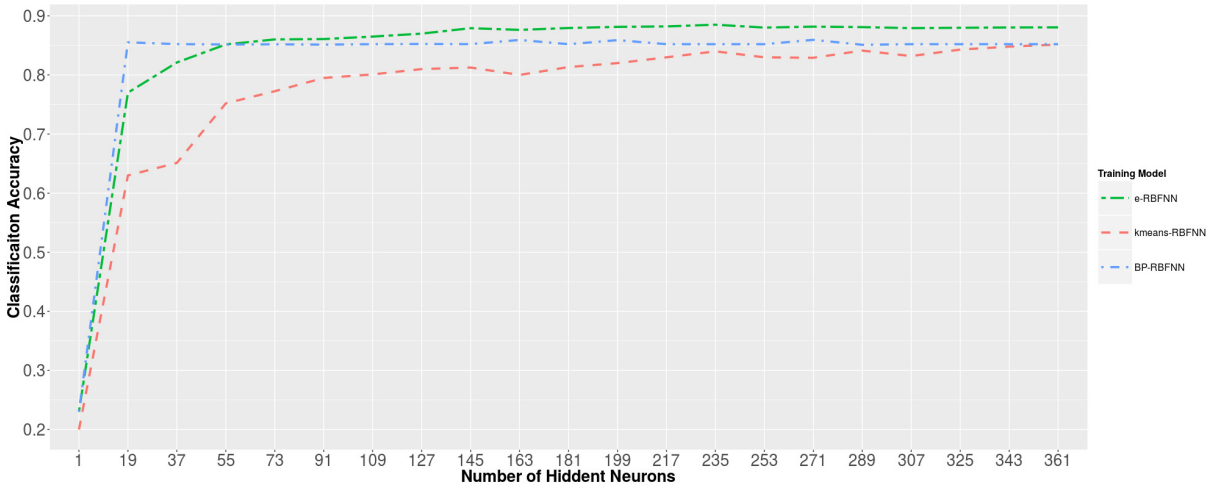
The second experiment focuses on a specific data set for examining the classification performance by RBFNNs with different number of hidden neurons. The RBFNNs trained via three different training schemes and their classification performances are compared. The e-RBFNNs employ the eigenvectors to determine the RBF centers. The k-means RBFNNs adopt the RBF centers determined by k-means. Besides these two two-phase trained RBFNNs, the classification accuracies of RBFNNs with all parameters trained via back-propagation (BP-RBFNN for short) simultaneously and iteratively are also taken for the results comparison. For demonstrating the efficiency and reliability of each RBF center determining approach, the performance is evaluated from two aspects, the training time and the classification accuracy.

The data set used in this experiment is the Landsat Satellite Image data set, which contains 6435 image samples, and each sample has 36 features. The samples are labeled as 7 classes. A 5-fold cross-validation is adopted to test the precision statistically. What is more, to eliminate the effect of different scaling of each feature, all the input feature values are normalized before the experiment, i.e. A pre-processing of rescaling the range of features in $[-1, 1]$ is given to the data set.

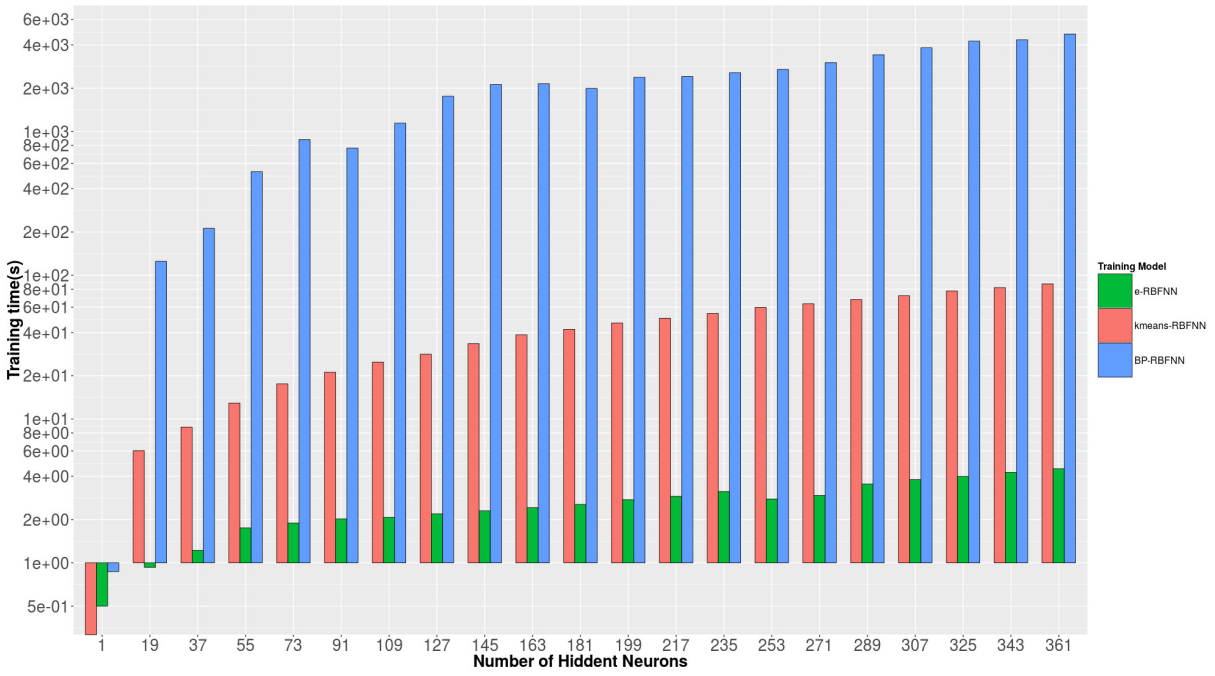
Generally, for an NN model, the number of hidden neurons is a key parameter that may affect its learning ability [92]. Obviously there is a monotonic increasing relationship between the number of hidden neurons and the training time of network. Therefore, the performances of different RBFNNs with increasing number of hidden neurons are compared in this experiment. Specifically, the number of hidden neurons in each RBFNN model is added from 1 to 361 with a step of 18 hidden neurons for enlarging

the network scale. While increasing the number of hidden neurons, the training time bars and output accuracy curves of the differently trained RBFNNs are drawn for the comparison. Fig.4.3 shows the comparison results of the employed models. Note that in Fig.4.3(b), for the training time comparison, since the orders of magnitude among different models are quite different, the y-axis is plotted on the logarithmic scale.

In Fig.4.3(a) the average accuracy curves shown from the results demonstrate several facts. All of the three differently trained RBFNNs can provide accuracies higher than 80%. Specifically, from the accuracy aspect, when the network scale is small (the number of hidden neurons is less than 55), the BP-RBFNNs offer higher accuracies than the other two models. However, the accuracy of the proposed e-RBFNN soon surpasses the accuracy of BP-RBFNN when the network scale becomes larger. In most of the time the k-means RBFNN gives the poorest accuracy among these three differently trained RBFNNs. From the training time aspect, as shown in Fig.4.3(b), the e-RBFNN takes the least training time. Combining the two sub-figures together, the BP-RBFNN shows a tiny decrease of accuracy when the network scale becomes larger, and it takes much longer training time than the other two. This shows the fact that directly training all the parameters of RBFNN simultaneously and iteratively may lead the model to a poor local minima [68]. Such result reflects the reason why the back-propagation training scheme is seldom employed in the real applications. Although the k-means RBFNN takes less training time than the BP-RBFNNs, the classification accuracies of k-means RBFNNs are the lowest at the beginning. However, the accuracy curve of the k-means RBFNN in Fig.4.3(a) presents a continuous increasing trend and finally surpasses the accuracy curve of BP-RBFNN. The efficiency and reliability of the eigenvector-based center determining approach is exhibited by the performance of e-RBFNN. The e-RBFNN provides higher accuracy while taking significantly less training time than that of the other



(a) The comparison results of average output accuracy



(b) The comparison results of average training time *

Figure 4.3. The comparison results of RBFNN via different training schemes on the Landsat Satellite Image data set

* Please note that since the y-axis is given a logarithmic transformation, the scale of y-axis for Fig.4.3(b) is uneven.

Table 4.2. Benchmarking data sets

Date sets	# Features	#Samples	#Classes
Thyroid Gland	5	215	3
Iris	4	150	3
Wine	13	178	3
Mushroom	22	8124	2
Segmentation(Seg)	19	2310	7
Satellite	36	6435	6
Ring	20	7400	2
Letter	16	20000	26
Sampbase	57	4597	2
MNIST	784	70000	10

two RBFNNs.

4.3.3 Experimental Results on Benchmarking Data Sets

More experiments have been conducted with other types of data for studying the performance of the proposed approach further. Benchmarking data sets from different domains with different characteristic are selected. The employed data sets include MNIST that the number of samples and the number of features are both relatively larger, and Letter and Mushroom consist of a relatively larger number of samples. In addition, small data sets and medium-sized data sets are also employed for the testing. The performance of RBFNN using the randomly generated RBF centers (R-RBFNN) [70, 71] is compared to substantiate the reliability of the proposed center determining approach. Besides, the RBFNN with centers determined via LVQ approach (LVQ-RBFNN), the RBFNN with centers determined via the Fisher Ratio (Fisher-RBFNN), the SV training approach and KLT scaled eigenvectors (KLT-RBFNN) are also added in the comparison. LVQ3 [93] is employed for the LVQ-RBFNN. Additionally, the RBFNN trained after a PCA transformation (PCA-RBFNN) is tested afterward. Detailed information of the employed data sets is given in Table 4.2.

Table 4.3 and Table 4.4 give the comparison results of experiment on these data sets by 5-fold cross-validation. According to the results in Table 4.3, the classification accuracy of each model does vary on different data sets. In 5 of the 10 data sets, the e-RBFNN trained gives the highest accuracy. In 4 of the 10 data sets, the e-RBFNN trained via the proposed approach offers the second or third highest accuracy. Compared with the mainstream k-means RBFNN, the difference between the classification accuracy obtained via these two training schemes is less than 2%. The proposed RBF center determining approach can save at least 80% of the average training time compared with the k-means RBF center selection approach. e.g., for the satellite image data set, when the number of hidden neurons was set as 361, it only took 4.74 seconds on average to train up an e-RBFNN. In the meanwhile, it costs 84 seconds on average to train up a classical k-means RBFNN. Similar to the results in Section 4.3.2, although the BP-RBFNN can provide higher training accuracy than the e-RBFNN for 2 of the 10 data sets, it needs much longer training time than others. The SV model gives the highest accuracy in 3 of the 10 data sets, but is shown much more time-consuming than the e-RBFNN. In general, the e-RBFNNs offer the comparable (even better) performance and need much less training time when compared with k-means RBFNN, SV, and BP-RBFNN.

More significant results come from the comparison between R-RBFNN that uses the randomly generated RBF centers and the e-RBFNN. Earlier studies reported that an RBFNN can be quickly trained up and provide acceptable accuracy when the RBF centers are randomly generated, therefore the R-RBFNN could be employed in some scenarios [70, 71]. This is consistent with the experimental results (see the corresponding columns in Table 4.3 and Table 4.4). Due to the fact that randomly generating RBF centers is much more time-saving than calculating the centers with other algorithms, it

spent the least time on training up an R-RBFNN in the experiment. However, the R-RBFNN offered the lowest classification accuracies in the experiment. For all the data sets used here, the e-RBFNNs can offer more than 3% higher accuracies than the R-RBFNNs. Especially, for the Satellite data sets, the accuracy enhancement of e-RBFNN to the R-RBFNN has reached to 8.1% high. More importantly, the output of the R-RBFNN is not stable enough. The results in Table 4.3 show that the R-RBFNN always has higher standard deviation of the classification accuracy than that of other methods.

The comparison results between the proposed model and the KLT-RBFNN model should also be noticed. Despite the slight difference between the KLT and PCA, eigenvectors play the key role in both of the two models. Compared with the KLT-RBFNN which directly adopts the eigenvectors as the centers, the proposed model employs the eigenvectors as the clustering indicators to give a relax solution of the k-means. The obtained centers of KLT-RBFNN therefore may be located outside the data clusters, while as shown in Fig.4.2, the obtained centers by the proposed approach are usually located near to the k-means clustering centers in the feature space. According to the results in Table 4.3 and Table 4.4, in the experiment, the KLT-RBFNN spends less time on finding the RBF centers, while the proposed model may offer higher classification accuracies for all the data sets.

Besides, the PCA-RBFNN, the LVQ-RBFNN and the Fisher-RBFNN have taken more training time while giving lower accuracy for almost all the employed data sets.

4.3.4 Fine Tuning Experiment

To make the proposed approach more convincing, a complementary experiment is included to evaluate the model from another aspect. As mentioned in Section 4.1.2,

Table 4.3. Accuracy (%) of different training schemes on benchmarking data sets

Data sets	BP-RBFNN	k-means RBFNN	R-RBFNN	e-RBFNN	PCA-RBFNN	SV	Fisher-RBFNN	LVQ-RBFNN	KLT-RBFNN
Thyroid Gland	84.26(1.47)*	84.75(1.95)	79.24(2.59)	**85.59(2.01)**	81.23(1.19)	84.01(1.26)	80.15(2.59)	82.33(0.47)	83.14(0.55)
Iris	98.12(0.52)	96.98(0.93)	96.66(1.25)	97.71(0.38)	96.13(1.01)	**98.25(0.61)**	95.12(1.13)	96.10(1.09)	96.09(0.92)
Wine	94.45(0.25)	92.91(1.22)	91.21(3.01)	**95.15(1.02)**	92.08(1.93)	94.34(1.10)	90.15(2.01)	93.07(1.14)	90.29(0.71)
Mushroom	99.12(0.30)	99.44(0.33)	98.25(1.12)	**99.53(0.14)**	98.94(0.46)	99.24(0.21)	98.97(0.45)	99.23(0.19)	95.34(1.02)
Seg	72.17(2.46)	79.63(3.08)	73.32(4.01)	79.29(3.02)	73.09(3.58)	**80.23(2.82)**	72.25(3.01)	76.97(3.53)	70.92(1.17)
Satellite	85.39(2.21)	85.67(1.31)	79.47(3.27)	**88.16(0.97)**	83.35(1.97)	85.52(1.58)	82.73(1.84)	84.42(1.25)	80.21(0.88)
Ring	80.21(2.09)	82.15(2.13)	79.94(4.35)	**83.17(3.21)**	81.35(2.75)	81.89(2.29)	80.24(2.98)	79.95(2.12)	83.35(1.21)
Letter	80.27(2.17)	78.62(0.81)	73.39(3.94)	79.13(1.09)	76.32(1.57)	**81.43(1.15)**	78.12(1.12)	76.59(0.96)	74.87(0.31)
Spambase	87.72(1.15)	**89.35(1.05)**	85.52(3.35)	87.34(1.07)	86.10(1.21)	88.12(1.37)	84.35(1.54)	86.64(1.42)	85.53(1.39)
MNIST	95.25(1.32)	**97.51(0.71)**	93.18(3.32)	96.71(1.14)	93.21(1.17)	95.12(0.49)	96.35(0.79)	94.08(1.02)	92.01(2.27)

* The values in parentheses are standard deviations.

Table 4.4. Average training time (seconds) of different training schemes on benchmarking data sets

Data sets	BP-RBFNN	k-means RBFNN	R-RBFNN	e-RBFNN	PCA-RBFNN	SV	Fisher-RBFNN	LVQ-RBFNN	KLT-RBFNN
Thyroid Gland	0.6066	0.3233	$\approx$ **0**	$\approx$ **0**	0.4843	1.7012	0.6724	0.3212	$\approx$ **0**
Iris	0.0230	0.0015	$\approx$ **0**	$\approx$ **0**	0.0129	0.0207	0.0228	0.0027	$\approx$ **0**
Wine	0.0461	0.0240	$\approx$ **0**	$\approx$ **0**	0.0357	0.0439	0.0397	0.0218	$\approx$ **0**
Mushroom	0.59e+03	147.39	**2.43**	18.45	204.22	0.49e+03	0.52e+03	189.42	15.21
Seg	5.31e+02	61.27	**1.42**	6.45	75.52	3.92e+02	4.14e+02	50.35	5.12
Satellite	5.60e+03	84.24	**3.12**	4.74	103.72	4.53e+03	4.34e+03	94.52	4.40
Ring	1.87e+03	241.30	**14.85**	57.28	309.27	1.53e+03	1.67e+03	247.52	49.33
Letter	9.35e+03	1.63e+03	**88.86**	254.52	1.57e+03	9.25e+03	8.98e+03	1.49e+03	241.98
Spambase	3.32e+03	234.45	**5.32**	30.14	307.12	2.75e+03	2.42e+03	197.56	28.81
MNIST	8.57e+05	1.89e+05	**437.24**	0.39e+04	2.21e+05	6.72e+05	5.13e+05	1.55e+05	0.37e+04

in some applications, a fine-tuning phase is added after a two-phase training scheme. Accordingly, a fine-tune phase is added on the e-RBFNN and the most widely employed k-means RBFNN. The results are given in Table 4.5.

As shown in Table 4.5, it demonstrates that for most of the data sets, a fine-tuning phase can bring a tiny enhancement of the output accuracy. Such operation has also been widely employed in multilayer network training recently [68]. For both of the e-RBFNNs and k-means RBFNN, accuracy enhancements are observed after the fine-tuning phase. Meanwhile, the average accuracy improvement of the e-RBFNNs is 1.6%, the average accuracy improvement of k-means RBFNNs is less than 1.0%. These complementary experimental results shall be considered as another advantage of the pro-

Table 4.5. The results after fine-tuning

	k-means RBFNN		e-RBFNN	
	Accuracy	Ave.Acc*.improvement	Accuracy	Ave.Acc.improvement
Thyroid Gland	85.12	0.37	87.22	1.63
Iris	97.33	0.35	97.82	0.11
Wine	95.25	0.80	96.98	1.83
Mushroom	99.50	0.06	99.61	0.08
Segmentation	79.97	-0.15	80.35	2.18
Satellite	87.37	1.70	89.99	1.83
Ring	85.25	3.10	85.54	2.37
Letter	80.35	1.73	81.98	2.85
Spambase	90.12	0.77	90.01	2.67
MNIST	96.99	0.74	97.20	0.49

* Ave.Acc. is for Average Accuracy. This experiment is also conducted via 5-fold cross validation.

posed model.

In summary, the eigenvector-based RBF center determining approach is compared with others for the appraisal. The comparing schemes include both the unsupervised and supervised approaches as discussed in Section 4.1.3. For the unsupervised approaches, the k-means clustering, which is the most widely employed one in real applications was adopted as the comparing objective. On the other hand, the supervised approaches can be regarded as another branch for the RBF center calculation. Accordingly, several supervised approaches are also compared. Besides, the R-RBFNN which adopts the randomly generated RBF centers is also considered in the comparison. In Table 4.6, as a summary, it presents the comparison of the time complexity of calculating the RBF centers, the output accuracies, and some important features of these approaches.

From the summarized results in Table 4.6, the proposed eigenvector-based approach has the second fastest training speed. Moreover, as shown in the experiments, this approach can help the RBFNNs offer higher accuracy than that of the random RBF center selection approach. When compared with other two catalogs of approaches, the proposed approach presents an advantage on the time complexity while providing a comparable reliability. It demonstrates that the eigenvector-based approach can provide

Table 4.6. Summary about the performance of different approaches

	Time complexity	Accuracy	Important feature
Random-center	$O(1)$	poorer than others	Only need to calculate the weights Not reliable
Supervised Approaches	$O(nd^2DLt)$	high for small-scale net	Optimizing the entire network according to the labels Easily poor local minima
Unsupervised Approaches	$O(ndDt)$	high for large-scale net	Fully pre-training Accuracy improved after fine-tuning Rapid convergence of the RBFNN
Proposed Approaches	$O(d^3)$	comparable with others	A relax pre-training and Accuracy improved after fine-tuning Rapid convergence of the RBFNN

* Note that in the first column, t is the number of iterations.

a better balance between the time and accuracy consideration for determining the RBF centers for an RBFNN.

4.4 Conclusions

In this chapter, a fast eigenvector-based RBF center determining approach for RBFNN is proposed. To improve the training speed of RBFNN reliably, first, the RBF centers is identified via eigenvectors in the feature space, then the output weight can be easily obtained with the pseudo-inverse solution or gradient descent approach. Experimental results have shown that the training time of the RBFNN is greatly reduced when the eigenvector-based RBF center determining approach is employed. The classification accuracy is maintained, or even improved for certain data sets. The results also show that the proposed approach is more reliable than fast randomly-generating approach for determining the RBF centers. Thus we can readily conclude that the proposed model provides a better balance between the training time and the reliability for training up an RBFNN.

However, this work still has limitations. The data sets employed in the experiment are all relatively balance distributed data sets, and the distribution of the given data may affect the choice of the RBF centers. Therefore the incorporation of experiments on imbalanced data sets with RBFNN shall offset such limitation. Moreover, from the theoretical aspect, some of the results, e.g., the results of the fine-tuning experiments shall require mathematical explanation. In the future, we will develop this study further by concentrating on these two problems.

Chapter 5

Feature Learning for Multiple Domain Data

As discussed in previous chapters, multi-layer NNs have great advantages in feature representation, especially for large scale image data set. Most of the state-of-the-art results on multi-layer neural network models are reported focusing on the fields of computer vision, automatic speech recognition, and natural language processing. The great success of deep learning in these application scenarios demonstrates the advantages of using neural network approaches to learn a representation of a given data set. The superiorities of using a multi-layer neural network for feature learning include that neural networks are adept at processing the complex nonlinear relationships among the variables, and the multi-layer structure can take advantage of the increasing quantity of available data. However, in real applications of different domains, things often become more complicated, features usually have different characteristics with image data or NLP data, for example, there exists a time-series relationship among the features, or the

features stem from different domains. It is worth investigating the possible utilization of deep learning technology for feature representation on different applications.

This study explores the potential of feature representation with multi-layer neural networks on real weather data. The weather data employed in this study was hourly collected weather records in the past 30 years by the Hong Kong Observatory (HKO). These weather records are basically characterized by time series, univariate and continuously valued. Besides the basic features, the employed weather data sets also have some specialties. One important trait of these data sets is that there are hidden nonlinear relationships among various univariate data sequences, e.g., the temperature records may have some interaction with the wind speed records, therefore, a proper fusing representation of the related univariate data sequences may be helpful to improve the simulation accuracy of these data sets. Another specialty of the employed data set is that there is season-to-season and year-to-year variability in the trend of weather data. The great learning power of the multi-layer neural network is expected to capture these trends during the learning process.

This chapter focuses on the feature representation issue of the given weather data sets from multiple meteorological domains including the temperature data, the atmospheric pressure data and the wind speed data. In detail, firstly, several widely used multi-layer neural network architectures are tested on each kind of weather data set respectively, results in this stage of the study demonstrated the potential of multi-layer neural network for the feature representation on the time series univariate data sets. Consequently, to improve the forecasting accuracy of wind speed data in advance, the information from temperature data and atmospheric pressure data is utilized together with wind speed records to train up the forecasting models. Several models that can learn a fusing representation of time series data from different domains, including the

proposed canonical correlation analysis based Split-Autoencoder, are tested and compared. The employed computational intelligence models (e.g. Support Vector Machine) trained up in the learned fusing feature space are expected to provide higher prediction accuracy. The experimental results demonstrate that by considering the canonical correlation among multi-domain data sets, a fusing represented feature space can capture the relevant information from various domains. Forecasting models trained up in this fusing represented feature space provided the best performance in the experiments.

5.1 Preliminary

5.1.1 Weather Data and Weather Forecasting via Machine Learning Models

The changes of climate can greatly impact every aspect of people's life. In the wind energy industry the fluctuation of wind speed can guide the selection of the site position; engineers frequently utilize information based on wind speed, pressure, temperature forecasts in the design and construction of large wind-resistant structures such as bridges, high-rise buildings, and off-shore oil platforms; even in financial markets, weather forecasting also plays a critical role as weather derivatives and the need to manage weather-related risks [94, 95]. Therefore, many significant research efforts are utilized to develop weather forecasting methods.

Generally, the Numerical Weather Prediction (NWP) models, which are based on physical principles and use complex mathematical models of the atmosphere and oceans to predict the weather, have occupied the dominate position in weather forecasting techniques [96, 97]. However, accompany with the great success of computational intelli-

gence obtained on many real applications in recent years [8, 98, 99], both research and industrial community are paying more attention to employing machine learning techniques for weather predicting, e.g., instead of buying weather information from observatories, some wind power plants have employed machine learning approaches to forecast the wind speed in next several hours [100]. Moreover, Machine learning approach has been considered as an important assistant method to improve/adjust the output of numerical models for meteorologists [101]. Compared with NWP models, machine learning approaches offer the advantages in lower requirement on domain knowledge and equipments such as meteorological instruments. Machine learning approaches, especially, Neural Networks (NNs), are adept at predicting the weather condition from historical weather records by exploring the complicate, nonlinear relationship among the given records [100].

The weather quantities involved in this study include the temperature, the atmospheric pressure, and the wind speed. Totally 30-year historical data records of these weather physical quantities are hourly collected from observation points in Hong Kong by the HKO. Hong Kong is characterized by a long coastline and numerous islands for such a relatively small territory. The mesoscale weather system of Hong Kong is quite different from other places since it is heavily affected by rainstorms and tropical cyclones [101], moreover, the high building density may also affect the weather condition of Hong Kong. Therefore, finding the disciplines and capturing the possible cycles of wind speed change in Hong Kong is more difficult than other places in sub-tropical regions.

Weather prediction is a typical time series problem. For time series analysis, univariate time series regression is the most fundamental and most widely applied framework for quantitative value prediction [102]. Generally speaking, for a certain variable,

the objective of univariate time series regression is to find the relationship between its status at a certain future time point and its status at a series of past time points, and estimate its future status via

$$v_t = f(v_{t-\Delta t-1}, v_{t-\Delta t-2}, \ldots, v_{t-\Delta t-n}), \tag{5.1}$$

where Δt is the prediction horizon. For the prediction tasks on hourly weather data records. When $\Delta t \leq 3$ hours ahead, it is called nowcasting; normally, researchers focus on short-term forecasting tasks in which Δt is setting as 4 to 7 hours ahead; in some special cases, such as wind speed changing prediction for wind power plant to estimate the gap between energy supply and energy consumption, Δt is usually set to 12 hours ahead. A prediction task with a longer prediction horizon usually has lower prediction accuracy. The univariate time series regression framework is quite effective for short-term forecasting [103], but for long-term forecasting tasks, NWP model is more recommended.

As discussed above, machine learning models are more and more commonly employed in weather forecasting area, the function f, can be obtained by employing different machine learning models such as Linear Regression, Generalized Linear Model, Auto Regressive Integrated Moving Average Mode (ARIMA) [104,105]. Many previous studies have discussed the strengths and weaknesses of the different machine learning models to process the weather data [106, 107].

5.1.2 Multi-layer Neural Network for Feature Representation

Although the idea of multi-layer neural networks has been proposed for more than two decades, it wasn't widely used until Hinton's research in 2006 [99]. The essential

challenge in training up a multi-layer architecture is to deal with the strong dependencies that exist during training between the parameters across layers [68]. Multi-layer neural networks usually have more parameters than those with shallow architectures. Moreover, in a multi-layer neural network architecture, due to the non-convexity of the complex model, the optimization with traditional Back-Propagation training approach may fall in a local minimum rather than a global minimum. This may bring poor generalization to the model.

The Deep Belief Network (DBN) that greedily trained up one layer with a Restricted Boltzmann Machine (RBM) at a time was introduced in 2006 [99]. Starting from this work, researchers solved the training problem of deep neural networks in two phases. In the first phase, unsupervised pre-training, all layers are initialized using this layer-wise unsupervised learning signal; in the second phase, fine-tuning, a global training criterion (a prediction error, using labels in the case of a supervised task) is minimized [108]. Fig.5.1 gives the architecture of a typical DBN for illustration of the Multi-layer Neural Networks. As shown in Fig.5.1, by stacking multiple layers of simple learning blocks, the hierarchical architecture of the neural system can represent the input raw data according to the interactions of many complicated factors on multiple levels [92].

5.2 Feature representation for Univariate Weather Data with Multi Layer Neural Networks

In this section, several widely used multi-layer neural network architectures are employed for feature representation of each kind of weather data set respectively. The

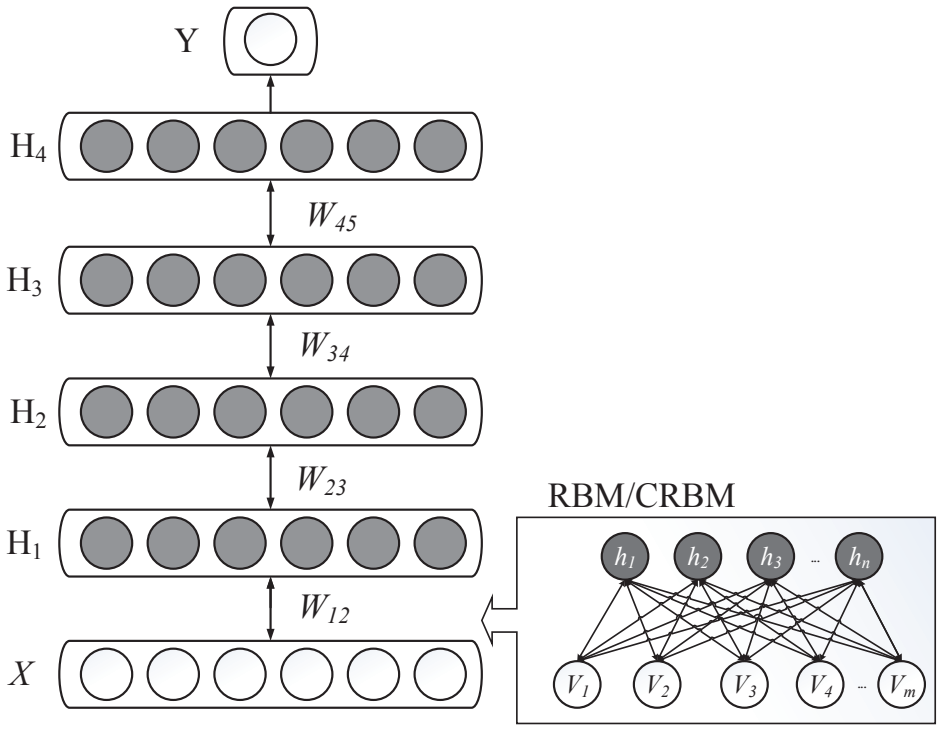


Figure 5.1. A typical DBN with 4 hidden layers, each layer is a simple RBM block. The output of each intermediate layer can be viewed as a representation of the original input data.The RBM block can be substituted by other type of neuron layer (e.g. Autoencoder) depending on the learning objective.

purpose of the section includes (1) exploring the potential of multi-layer neural network for feature learning on univariate time-series data; (2) improving the weather forecasting accuracy for different prediction horizons; (3) making the comparison among the performances of different multi-layer networks on the given weather data.

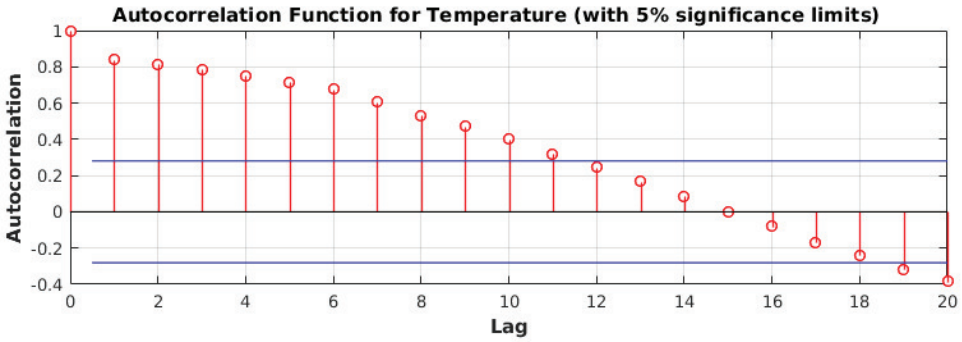
5.2.1 Data Preparation

Historical weather data sets, including the temperature, Mean Sea Level Pressure (MSLP) and wind speed data are employed in our model. The time range of the data sets is almost 30-year long, which covers the period from January, 1st, 1983 to December, 31st, 2012. In detail, the numbers of temperature, MSLP, and wind speed records are more than 260,000 respectively. The temperature (measured in degree Celsius) and MSLP (measured in hectopascal(hPa)) data are scalar data sets. On the contrast, the wind speed data has two dimensions: the polar coordinate for the wind direction (measured in degree angle) and the speed (measured in meters per second). Moreover, at some time points, the directions of the air motion are not stable, i.e. the wind directions at these time points are not fixed. such kind of data is considered as missing value in this study. Therefore, some pre-processing on the data sets is necessary. In detail, for the wind speed data, this study tries to predict the components of the air motion in 0-degree angle direction. Thus, by denoting the angle as θ and the speed as v, the raw data can be transformed via

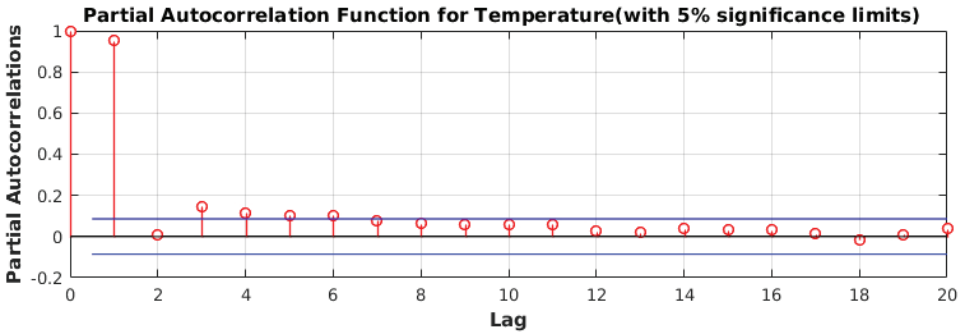
$$v^0 = \cos\theta \cdot V, \tag{5.2}$$

where v^0 is the vector components of the wind speed in 0-degree angle direction.

To train up a neural network model with time series data, it is necessary to know the relation that exists between the series and their lags. In some previous studies, the lags are determined by expert experience. In this study, statistics approaches are employed to decide the input variables. Specifically, the input variables of the neural network are determined by two measures: autocorrelation function (ACF) and the partial autocorrelation function (PACF) [109, 110]. For example, Fig. 5.2, Fig.5.3 and Fig.5.4 illustrate the ACF plots and PACF plots for the hourly observed temperature, pressure



(a) ACF plots for the temperature data



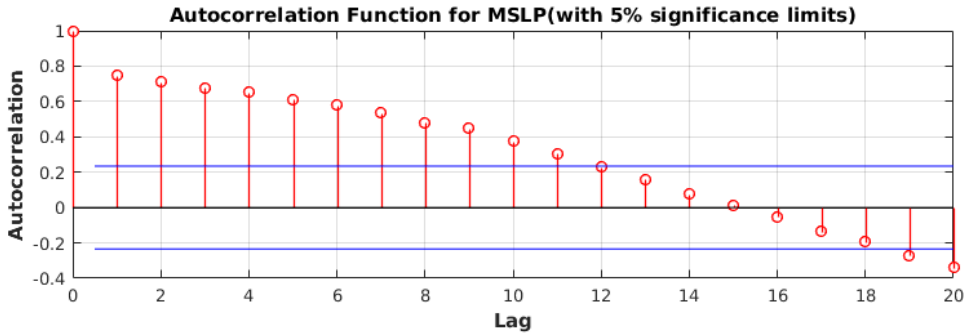
(b) PACF plots for the temperature data

Figure 5.2. ACF & PACF plots for the temperature data in Hong Kong

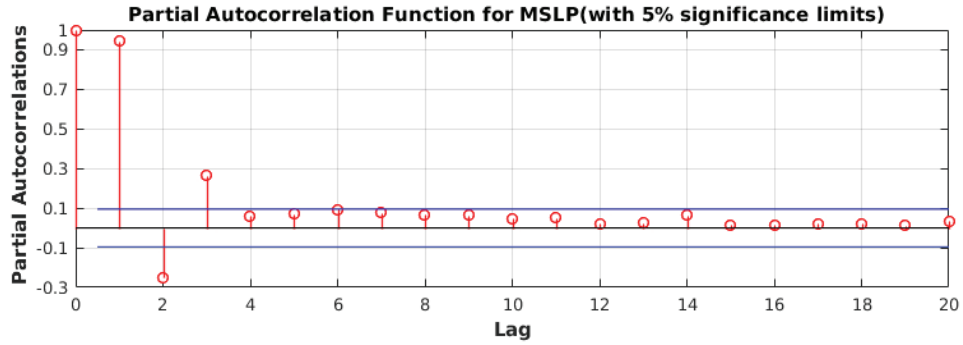
and wind speed records in the entire data sets when $\Delta t = 3$.

5.2.2 Weather Feature Learning with Autoencoder

A typical Autoencoder tries to learn a function $\underset{w,b}{\mathbf{h}}(x) \approx x$. In other words, it is trying to learn an approximation to the identity function, so as to output $\hat{x}$ that is similar to x. By placing constraints on the network, such as by limiting the number of hidden units, some interesting structure about the input data may be discovered [111]. If there



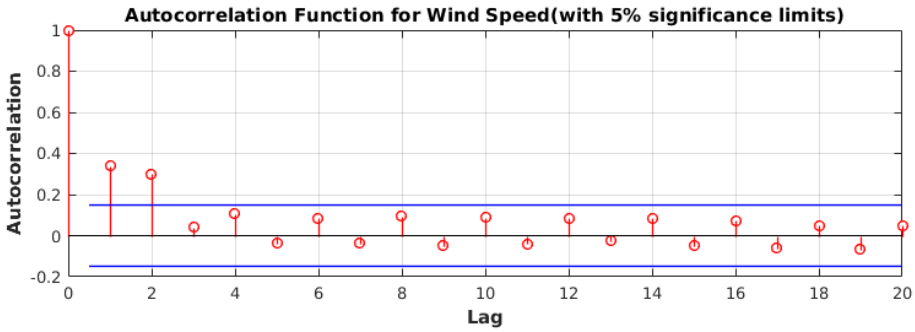
(a) ACF plots for the MSLP data



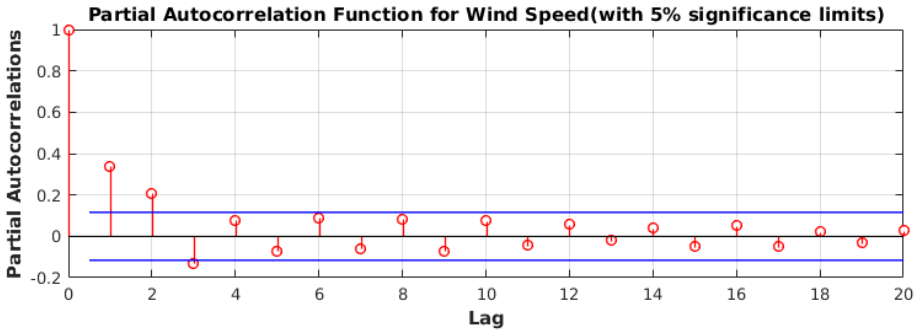
(b) PACF plots for the MSLP data

Figure 5.3. ACF & PACF plots for the MSLP data in Hong Kong

is a certain structure hidden in the data, for example, if some of the input features are correlated, such as in the feature space of time series analysis, this model may discover some of those correlations. An illustration of a typical Autoencoder layer is shown in Fig.5.5.



(a) ACF plots for the wind speed data



(b) PACF plots for the wind speed data

Figure 5.4. ACF & PACF plots for the wind speed data in Hong Kong

The loss function of Autoencoder is

$$J(W,b) = \left[\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}\left\|\mathop{\mathbf{h}}_{W,b}(x^{(i)}) - x^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2, \tag{5.3}$$

where N is the number of training samples. The objective of the Autoencoder is to minimize Eq.5.3 in order to make sure that the output $\mathop{\mathbf{h}}_{W,b}(x^{(i)})$ can approximate the raw data $x^{(i)}$ as far as possible. The second term in Eq.5.3 is a regularization term (also

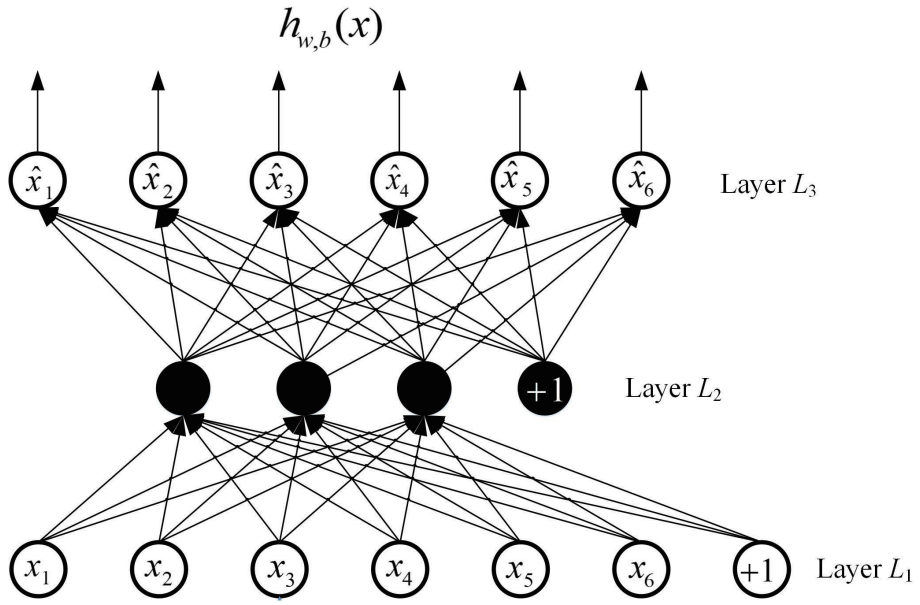


Figure 5.5. An illustration of Autoencoder Algorithms. Layer L_1 is the input layer, and L_3 is the output layer (or reconstruction layer). Via hidden layer L_2 (or representation layer), the input x in layer L_1 is represented in a new feature space, and the output $\hat{x}$ in L_3 is expected to approximate x.

called a weight decay term) controlled by the weight decay parameter λ that tends to decrease the magnitude of the weights, and helps prevent overfitting.

Consequently, several Autoencoders are connected layer by layer with a stacked structure to build the deep neural network. Specifically, in the training process of each layer, as shown in Fig.5.5, the input vectors have to pass through the three layers, and the vectors in hidden layer (layer L_2 or representation layer) are the representations of the input vectors and can be used to reconstruct the input vectors in layer L_3 (the reconstruction layer). Thus, in every layer of the stacked Autoencoder, the input of the current layer is the output of the previous layer, and the output is the transformed vectors in the L_2 of the current layer.

5.2.3 Weather Feature Learning with Continuous RBM

RBM is another important structure to model a deep neural network structure. The RBM is a two-layer network with one visible layer and one hidden layer. Fig.5.6 gives an illustration of RBM architecture. As shown in Fig.5.6, the standard type of RBM has binary-valued (Boolean/Bernoulli) m hidden and n visible neurons, and consists of a matrix of weights $W = (w_{i,j})$ (size $m \times n$) associated with the connection between hidden neurons h_j and visible neuron v_i, as well as bias weights (offsets) a_i for the visible units and b_j for the hidden units. The word "restricted" means that there is no connection between any two neurons in the same layer. The energy function of a configuration (pair of boolean vectors) (v, h) is defined as

$$E(v, h) = -\sum_i a_i v_i - \sum_j a_j v_j - \sum_i \sum_j v_i w_{ij} h_j. \tag{5.4}$$

The probabilities of the states of the visible and hidden neurons can be obtained via the sigmoid function

$$p_{vi} = p(vi = 1) = \frac{1}{1 + exp(-\sum_i w_{ij} h_j)} \tag{5.5}$$

and

$$p_{hj} = p(hj = 1) = \frac{1}{1 + exp(-\sum_i w_{ij} v_i)} \tag{5.6}$$

respectively.

An RBM architecture is trained to maximize Eq.5.7, which is the product of prob-

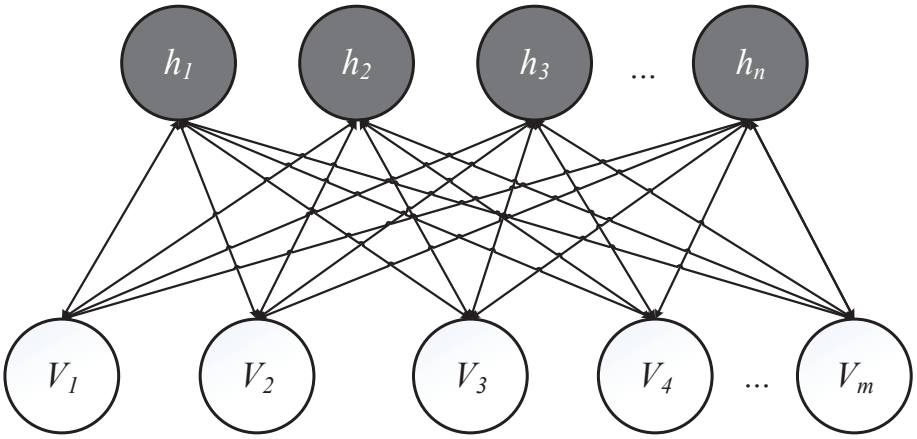


Figure 5.6. The typical architecture of a classical RBM model with two layers, m neurons in the visible layer and n neurons in the hidden layer, all neurons a binary-valued and no connection between any two neurons in the same layer.

abilities assigned to some training set V:

$$\arg\max_{W} \prod_{v \in V} \frac{1}{Z} \sum_{h} e^{E(v,h)}, \tag{5.7}$$

where the term $\frac{1}{Z} \sum_h e^{E(v,h)}$ is the probability distributions over hidden and/or visible vectors. Eq.5.7 can be optimized via the Minimising Contrastive Divergence (MCD) training rule by updating the weight value w_{ij} in each iteration according to

$$\Delta w_{ij} = \varepsilon(v \cdot h^{\mathrm{T}} - \hat{v} \cdot \hat{h}^{\mathrm{T}}), \tag{5.8}$$

where $\hat{v}$, $\hat{h}$ is the reconstructed states of the node in the last iteration [26].

A typical RBM is employed for binary valued data. In this work, a revised version of RBM, the continuous RBM (CRBM) is introduced for the feature learning of the given weather data sets [112]. The continuous stochastic neurons, which have the form

in Eq.5.9, are employed to take the places of the binary-value neurons by adding a zero-mean Gaussian noise to the input of a sampled sigmoid neuron.

$$s_j = \varphi_j \cdot (\sum_i w_{ij} s_i + \sigma \cdot N_j(0,1)) \tag{5.9}$$

with

$$\varphi_j(x_j) = \theta_L + (\theta_H - \theta_L) \cdot \frac{1}{1 + exp(a_j x_j)}, \tag{5.10}$$

where $N_j(O,1)$ represents a Gaussian random variable with zero mean and unit variance. The constant σ and $N_j(O,1)$ thus constitute a noise input component $n_j = \sigma \cdot N_j(O,1)$ according to a probability distribution

$$p(n_j) = \frac{1}{\sigma\sqrt{2\pi}} exp(\frac{-n_j^2}{2\sigma^2}). \tag{5.11}$$

The parameters θ_L, θ_H and a_j control the asymptotes and slope of the sigmoid function of each neuron. By this way, the nature and extent of the neurons stochastic behavior is simulated as the noisy units [113].

Consequently, the energy function of CRBM is analogous to that of the continuous Hopfield model as

$$E_{CRBM} = -\frac{1}{2} \sum_{i \neq j} w_{ij} s_i s_j + \sum_j \frac{1}{a_j} \int_0^{s_j} \varphi^{-1}(s) ds. \tag{5.12}$$

By using the MCD rule, in each iteration, parameters in CRBM model can be updated via

$$\Delta w_{ij} = \varepsilon_w (s_i \cdot s_j^{\mathrm{T}} - \hat{s_i} \cdot \hat{s_j}^{\mathrm{T}}) \tag{5.13}$$

and

$$\Delta a_j = \frac{\varepsilon_a}{a_j^2}(s_j \cdot s_j^{\mathrm{T}} - \hat{s_j} \cdot \hat{s_j}^{\mathrm{T}}). \tag{5.14}$$

5.2.4 Experimental Results for the Weather Data Sets

Machine learning forecasting models are expected to provide different performances in different feature spaces. In this study, a series of experiments were conducted to test if a performance improvement can be obtained when employing the forecasting model on the learned feature space. Specifically, for each of the temperature data, the MSLP data, and the wind speed data, the forecasting models were trained up via the original feature space, the feature space learned via Antoencoder and the feature space learned via CRBM respectively. To make the results more convincing, in the experiments, Δt was alternately set as 3 hours (nowcasting), 7 hours (short-term forecasting) and 12 hours in each forecasting task.

In this group of experiments, the Support Vector Machine Based Regressor (SVR) is employed as the forecasting model. Previous studies have reported that SVRs usually have higher generalization when compared with other machine learning forecasting models in weather forecasting, and widely employed to deal with univariate forecasting problems in practice [114–116]. SVR is a reliable choice as the forecasting model to evaluate the represented features in this study.

In detail, for each data set, the SVRs were trained up in three different feature spaces respectively. As in most of the previous studies [114–116], for univariate time series forecasting, the initial feature space is composed of the previous n hourly status $v_{t-\Delta t-1}, v_{t-\Delta t-2}, \ldots, v_{t-\Delta t-n}$, where n is determined by the ACF and PACF [110]. In the experiment, the initial feature space was represented via Autoencoder and CRBM re-

Table 5.1. The temperature prediction results by SVRs trained in different feature spaces

	SVR		SVR in AE		SVR in CRBM	
	MSE	R^2	MSE	R^2	MSE	R^2
$\Delta t = 3$	0.2246	0.9919	0.1651	0.9940	1.2512	0.9332
$\Delta t = 7$	0.2304	0.9918	0.1820	0.9934	1.8512	0.9121
$\Delta t = 12$	0.3524	0.9873	0.1930	0.9930	2.1311	0.8754

Table 5.2. The MSLP prediction results by SVRs trained in different feature spaces

	SVR		SVR in AE		SVR in CRBM	
	MSE	R^2	MSE	R^2	MSE	R^2
$\Delta t = 3$	0.1483	0.9965	0.0983	0.9977	1.2721	0.9412
$\Delta t = 7$	0.1932	0.9956	0.0964	0.9977	1.5523	0.9313
$\Delta t = 12$	0.2135	0.9924	0.1472	0.9961	2.0012	0.9094

spectively. The forecasting performances of SVRs trained up in the three feature spaces were compared. To lower the effect of SVR parameters to the forecasting results, the parameters of SVRs in each feature space were same configured. For each prediction task, 80% samples are randomly chosen from the whole data sets for training, and the remaining 20% records are employed for the testing. As in previous studies, the forecasting performance is evaluated via the mean squared error (MSE) and the coefficient of determination, which is represented as R^2 value. The experimental results are given in Table.5.1, Table.5.2 and Table.5.3 respectively. Please note that in these tables, SVR, SVR in AE and SVR in CRBM means SVR in the original feature space, SVR in the Autoencoder learned feature space and SVR in the CRBM learned feature space respectively.

Table.5.1 and Table.5.2 present the forecasting results of temperature data and pres-

Table 5.3. The wind speed prediction results by SVRs trained in different feature spaces

	SVR		SVR in AE		SVR in CRBM	
	MSE	R^2	MSE	R^2	MSE	R^2
$\Delta t = 3$	1.5925	0.7811	1.5132	0.7903	1.9102	0.6994
$\Delta t = 7$	2.0213	0.7141	1.9008	0.7224	2.2332	0.6547
$\Delta t = 12$	4.4036	0.6124	4.1041	0.6435	5.5043	0.5227

sure data respectively. In real applications, the short-term prediction for these two kinds of data gives little challenge since that their variations in a short period are relatively stable and significant periodic trends can be observed [117]. This is also demonstrated by the results shown in Table.5.1 and Table.5.2. For example, for 3-hour ahead horizon forecasting of temperature, in the experiment, the R^2 was up to 0.9919 while the MSE was low to 0.2246 when the employed SVR was naively trained up in the original feature space. It can also be observed that the forecasting accuracy had a slight decrease when Δt was set longer. However, even the prediction horizon was set to 12 hours ahead, the R^2 was close to 0.99 while the MSE was lower can 0.4. The SVRs trained on the Autoencoder learned feature space provided even better performances. Improvement of the prediction performance, although very slight, can be observed when Autoencoders were adopted to represent the features for all prediction horizons. The average enhancement of R^2 was 0.0031, while the average reduction of the MSE was 0.0710. Table.5.1 also gives the forecasting performance of SVRs trained on the feature spaces learned via CRBMs. In contrast of SVRs trained up via the Antoencoder learned features, SVRs trained up via the CRBMs learned features cannot offer higher prediction accuracies. Conversely, feature representation via CRBMs was demonstrated that it leaded significantly lower forecasting accuracies and poorer generalizations for all of the three pre-

diction horizons. Specifically, the average R^2 decrease was about 0.5 while the average MSE increase was larger than 1.2. Similar results were observed in the MSLP data. As demonstrated in Table.5.2, compared with SVRs trained up with the original features, SVRs trained up in the Autoencoder learned feature spaces can provide a slight enhancement of the prediction accuracy and SVRs trained up with the CRBM learned features offered significantly lower forecasting accuracy and poorer generalization.

According to Table.5.1 and Table.5.2, for temperature data and pressure data, simply training up SVRs on the raw data set can obtain very reliable prediction results. The slight improvement by training up the models with Autoencoder learned features is not a significant contribution from the application aspect. From the academical aspect, such improvement demonstrates that for the univariate time series data, feature representation via Autoencoder is helpful to improve the performance of forecasting models. Conversely, the results demonstrate that CRBM is not suitable for the feature representation of the univariate time series data. The transformations of the raw data via CRBMs caused deterioration to the prediction accuracy in the group of experiments.

Table.5.3 shows the experimental results on wind speed data. Compared with temperature data and pressure data, the change of wind speed data is much more stochastic. For forecasting models, it is more difficult to learn the changing patterns of wind speed data. As shown in Table.5.3, the R^2 value of the wind speed prediction via SVRs trained up on the raw data set were 0.7811, 0.7141 and 0.6124 for Δt set as 3 hours, 7 hours and 12 hours respectively, which were much lower than the prediction accuracies obtained on the other two data sets. Similar with other two data sets, higher prediction accuracies were observed when Autoencoders were employed to learn a feature space for training up the SVRs. For different prediction horizons, the maximum R^2 increase was higher than 0.3. Moreover, SVRs trained up in the CRBMs learned feature spaces still provided

the poorest accuracies for the wind speed data, the R^2 values were lower than 0.7 for all the prediction horizons.

The series of the experiments demonstrate the fact that for the univariate time series forecasting, especially the weather data forecasting tasks, a proper representation of the raw feature is helpful to improve the performance of the employed machine learning models. Specifically, for the two popular feature learning tools which can be employed to process the univariate time series issues, the Autoencoder shows the potential to learn a feature space that may enhance the generalization of the employed forecasting model, while the CRBMs/RBMs were observed not very fit for this application scenario. This may be because of the structures of the two models. For Autoencoder model, according to Eq.5.3, the output of the model is also required to keep the correlations among the variables at different time points. The time series data could be reconstructed via linear/nonlinear transformations from the represented feature space, therefore actually the time-related information could be held in the training process of the Autoencoder. For the CRBM model, although the time series relationships among the variables may be considered in the training process, the continuous neurons, which are added to simulate the stochastic behavior, are very sensitive to the noises in the input signals, therefore this model is demonstrated not very suitable for the given scenario.

The experimental results also show that wind speed forecasting is much more difficult than the temperature and pressure forecasting. Short-term wind speed forecasting application has great significance from both academical and practical aspects. Based on the obtained experimental results, the following section will focus on using Antoencoder based model to improve the prediction accuracy of wind speed data further.

5.3 CCA-based Autoencoder for Cross-domain Feature Fusion of Weather Data

Previous experiments present two facts. First, the SVRs trained in the feature space learned via Autoencoder is helpful to enhance the prediction performance of the employed machine learning model. Second, the prediction accuracy of wind speed data have space to for improvement. In the following sections, Autoencoder based methods are proposed to improve the prediction accuracy of wind speed forecasting further.

In the previous section, the wind speed is predicted only based on its' previous status. If more factors that can affect the wind speed is employed as variables of the forecast model, the forecast accuracy may be improved. Meteorologists have revealed that there is complicated relationships between wind speed and air/earth surface temperature [118] as well as the pressure conditions [119]. For example, before a tropical storm the temperature often rises and the pressure usually become very low. It is reasonable to consider the temperature and pressure data as inputs for the forecasting model. Instead of naively put the temperature and pressure data into the forecasting model directly, in the following section, a revised Autoencoder model is proposed to fuse the time-series data sets from different domains by considering the correlations among different data sets.

5.3.1 Measure of Corrections: Canonical Correlation Analysis

In this study, the Canonical Correlation Analysis (CCA) is employed as a measure to analysis the correlation between two data sets. Set X', X'' are two equal-size data sets with different dimensions, $X' = (X'_1, \ldots, X'_N)^{\mathrm{T}}$ where $X'_i = (x'_1, \ldots, x'_{d_{X'}})$, $X'' =$

$(X_1'', \ldots, X_N'')^{\mathrm{T}}$ where $X_i'' = (x_1'', \ldots, x_{d_{X''}}'')$, $i = 1, \ldots, N$. to evaluate the corrections of the two data sets, simply calculate the correlation coefficient of every pair (x_i', x_j'') may neglect the inner correlation among dimensions in each data set. For time series data sets where the correlations among the dimensions (time points) has been identified by ACFs and PACFs as in 5.1.1, taking the inner correlations among dimensions in each data set into consideration is especially necessary. The CCA can evaluate the correlations of the two data sets by considering the dimensions of each data sets as a whole. Therefore, in this study, canonical correlation is adopted to analysis the correlations between two data sets.

The main concept of CCA is to calculate the correlation between a linear combination of the variables in one data set and a linear combination of the variables in the other data set. Specifically, CCA seeks projections vectors a_1, b_1 such that $y_1' = a_1^{\mathrm{T}} X'$ and $y_1'' = b_1^{\mathrm{T}} X''$ to maximize the correlation $corr(y_1', y_1'')$. y_1' and y_1'' are called the first pair of canonical variables. Consequently, CCA seeks projection vectors a_2, b_2 and to maximize $corr(y_2', y_2'')$ where $y_2' = a_2^{\mathrm{T}} X'$, $y_2'' = b_2^{\mathrm{T}} X''$, and subject to the constraint that $a_1^{\mathrm{T}} X'$ and $a_2^{\mathrm{T}} X'$ are uncorrelated, $b_1^{\mathrm{T}} X''$ and $b_2^{\mathrm{T}} X''$ are uncorrelated. This gives the second pair of canonical variables. Such procedure may be continued up to $\min\{d_{X'}, d_{X''}\}$ times. The first k pairs of canonical variables are $Y = (y_1', \ldots, y_k')$ and $Y'' = (y_1', \ldots, y_k'')$ and can be solved via finding the eigenvectors of a certain matrix (see the Appendix).

5.3.2 Cross domain feature fusion with Autoencoder

In the multi-domain feature learning scenario, the entire training feature matrix is composed of n multiple sub-feature matrices. Each sub-feature matrix can be considered as one view of the problem to be solved. Denote the entire feature matrix $X \in \mathbb{R}^{N \times d}$,

and X is composed of several sub-feature matrices $X^i \in^{N\times d_i}$, $\sum^{i} d_i = d$. Specific to the wind speed forecasting application scenario in this chapter, X is composed of three sub-feature matrices X^1, X^2 and X^3, where X^1, X^2, X^3 represent the feature matrix of historical wind speed data, historical temperature data, and historical pressure data respectively.

5.3.2.1 Directly Input

Directly using the feature matrix X as the input of the employed forecasting model is a popular solution for multi-domain data learning issue [120]. In some previous studies, machine learning models directly adopt the entire feature matrix is reported capable of providing satisfactory performance. In this work, the directly input strategy is employed for comparison. The performance of forecasting models that trained up in the feature space learned by the proposed model is expected to surpass the performance of forecasting model directly using X.

5.3.2.2 Autoencoder

In section 5.2.4, forecasting model trained up in the Autoencoder learned feature space was shown to be able to provide a provide better performance than that of trained up in the original feature space. According to this, typical Autoencoder is a possible strategy for the multi-domain data fusing representation. Specifically, this strategy will train a network by minimizing

$$J(W,b) = \left[\frac{1}{N}\left(\frac{1}{2}\left\|\mathop{\mathbf{h}}_{W,b}(X) - X\right\|^2\right)\right] + \frac{\lambda}{2}\|W\|^2. \tag{5.15}$$

Eq.5.15 is exactly same with Eq.5.3 in form. The architecture of the network is also same with the one illustrated in Fig.5.5. For this Autoencoder, the input X includes features from multiple domains. By minimizing Eq.5.15, the raw features from multiple domains are expected to be combined in the representation layer of the Autoencoder.

5.3.2.3 Split Autoencoder

One of the revised versions of Autoencoder for multi-domain data fusing representation is Split Autoencoder that is proposed in [121]. Different from typical Autoencoder that using the output in the reconstruction layer of the Autoencoder to approximate the entire input feature matrix, the SplitAutoencoder seeks to minimize the sum of the training errors of the sub-feature matrix from each domain. According to this strategy, a Split Autoencoder tries to learn a feature space from multi-domain inputs via minimizing

$$J(W,b) = \left[\frac{1}{N}\sum_{i=1}^{n}\left(\frac{1}{2}\left\|\mathbf{h}_{W_i,b}^{i}(X^i) - X^i\right\|^2\right)\right] + \frac{\lambda}{2}\|W\|^2. \tag{5.16}$$

Here W_i denote the weights connecting the corresponding input/output neurons w.r.t. X^i in the input/output layer and neurons in the shared representation layer. The motivation of this model is that the fused representation of the raw data from multiple domains can be used to reconstruct the sub-feature matrices of all domains. In the wind speed forecasting task, the wind speed data, the temperature data, and the pressure data are expected to be reconstructed from the learned feature space which is a shared representation of the X^1, X^2 and X^3.

Given a specific data set, Eq.5.16 could be efficiently solved via stochastic gradient descent (SGD) since the loss of this model is the empirical expectation of the loss

incurred at each training sample [121, 122].

5.3.2.4 CCA-Split Autoencoder

A further revision of Split Autoencoder is to take the correlations among the sub-feature matrices from different domains into consideration in the reconstruction layer of an Autoencoder. As discussed in Section 5.3.1, corrections between any two sub-feature matrices can be evaluated via the canonical correlation of these two matrices.

According to Section 5.3.1, for any two equal-sized matrices from different domains, CCA finds the pairs of linear projections of the two feature matrices that maximally correlated. To consider the canonical correlations in the Split Autoencoder model, for two sub-feature matrices X^i and X^j, terms related to the canonical correlation of their corresponding output $\tilde{X^i} = \underset{W_i,b}{\mathbf{h}^i}(X^i)$ and $\tilde{X^j} = \underset{W_j,b}{\mathbf{h}^j}(X^j)$ should be added in the objective function.

From the content of Appendix, for $k \leq \min(d_{X_i}, d_{X_j})$, denote $A \in \mathbb{R}^{k\times d_{X^i}}$ and $B \in \mathbb{R}^{k\times d_{x^j}}$ are the top k projections of $\tilde{X^i}$ and $\tilde{X^j}$. A and B can be identified via maximizing

$$\begin{aligned} &\mathrm{tr}(A^{\mathrm{T}}\Sigma_{12}B) \\ &\text{s.t. } A^{\mathrm{T}}\Sigma_{11}A = B^{\mathrm{T}}\Sigma_{22}B = I \end{aligned} \tag{5.17}$$

where Σ_{12} is the cross covariance between $\tilde{X^i}$ and $\tilde{X^j}$, Σ_{11} and Σ_{22} are the covariances of $\tilde{X^i}$ and $\tilde{X^j}$ respectively.

One solution for Eq.5.17 is to let U_k and V_k be the left- and right- singular vectors of $\Sigma_{ii}^{-1/2}\Sigma_{ij}\Sigma_{jj}^{-1/2}$ [123, 124]. Then the solution of A and B can be given as $\Sigma_{ii}^{-1/2}U_k$ and

$\Sigma_{jj}^{-1/2}V_k$ respectively and the objective value of Eq.5.17 is the sum to top k singular values of $\Sigma_{ii}^{-1/2}\Sigma_{ij}\Sigma_{jj}^{-1/2}$. After some mathematical transformation based on this, w.r.t. any two sub-feature matrices X^i and X^j, the canonical correlation term can be introduced into the objective function of the Split Autoencoder as

$$\frac{1}{N}\operatorname{tr}(U_k^{\mathrm{T}} \underset{W_i,b}{\mathbf{h}^i}(X^i) \underset{W_j,b}{\mathbf{h}^j}(X^j)^{\mathrm{T}}V_k). \tag{5.18}$$

As defined previously, in the wind speed forecasting task, X^1, X^2 and X^3 are the sub-feature matrices of wind speed, temperature, and pressure data. The objective function of the proposed CCA-split Autoencoder for the wind speed forecasting task thus can be given as

$$\begin{aligned} J(W,b) = &\left[\frac{1}{N}\sum_{i=1}^{3}\left(\frac{1}{2}\left\|\underset{W_i,b}{\mathbf{h}^i}(X^i) - X^i\right\|^2\right)\right] \\ &- \frac{\eta}{N}\left[\sum_{i=2,3}\operatorname{tr}(U_k^{1\,\mathrm{T}} \underset{W_1,b}{\mathbf{h}^1}(X^1) \underset{W_i,b}{\mathbf{h}^i}(X^i)^{\mathrm{T}}V_k^2)\right] \\ &+ \frac{\lambda}{2}\|W\|^2. \end{aligned} \tag{5.19}$$

There are three terms in Eq.5.19. The first term is the same with the term in Eq.5.16, which is employed to evaluate the error in the reconstruction of the three sub-feature matrices from the representation layer, and the third term is the conventional weight decay for the regularization. In the second term, since the objective of the whole task is to predict the wind speed, the canonical correlation between the wind speed and the temperature and the canonical correlation between the wind speed and the temperature are considered put into the objective function, and a parameter η is adopted to adjust the weights.

The principle under Eq.5.19 is to find a trade-off between the information captured in the representation layer mapping within sub-feature from each domain on the one hand, and the information in the relationship across the domains on the other hand. Intuitively, this is the same principle as the Information Bottleneck (IB) method [125]. According to the IB method, for two data sets X^i and X^j, the relevant information is that signal in X^i provides about signal in X^j. For a prediction task for X^i, the relevant information plays a role in the prediction. The IB is to squeeze the relevant information in X^j w.r.t X^i to extract an efficient representation for the further application. For Gaussian variables, CCA term in Eq.5.19 and the IB method actually involve the spectral analysis of the same matrices [126]. The CCA term in Eq.5.19 is applied to represent the relevant information that maximize the amount of the information about the wind speed while compressing the information about the temperature and pressures as much as possible. It therefore can be seen as a trade-off term to improve the generalization of the Antoencoder model.

Eq.5.19 could not be solved via SGD since the CCA term couples all training samples through the whitening constraints, however, a sufficiently large min-batch can be employed to optimize this objective effectively. Intuitively, this approach works because a large mini-batch contains enough information for estimating the covariances [127].

5.4 Experimental Results of Wind Speed prediction with Cross Domain Weather Data

In the previous section, several Autoencoder based cross domain feature fusion models are introduced including the proposed CCA-Split Autoencoder. In this section,

these models are employed to fuse the three different univariate time series signals, including the wind speed, the temperature, and the pressure. The data sets from different domains are expected to be represented in a learned feature space. The forecasting models trained up in the feature spaces learned via different approaches are adopted to predict the wind speed in the comparative experiments.

5.4.1 Data Preparation and Experiment Configuration

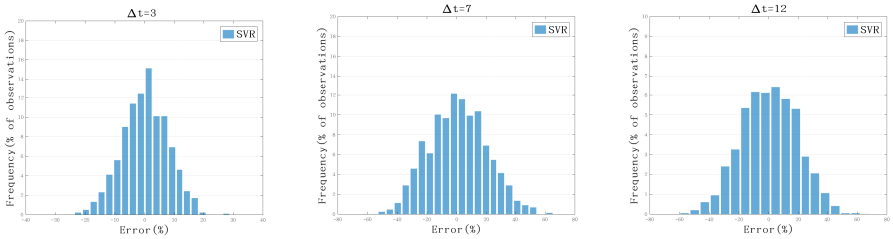
In this series of experiments, as discussed in previous section, the hourly temperature data, hourly MSLP data were adopted as input variables together with hourly wind speed data to train up the forecast model. Based on the ACF and PACF of wind speed data, the previous 36 hourly status of wind speed, temperature, and MSLP were employed as input variables, and the wind speed statuses at the next 3 hours, 7 hours and 12 hours were predicted. Specifically, the initially feature space is composed of previous $n = 36$ hourly records from the three different domains, $v^{temp}_{t-\Delta t-1}, \ldots, v^{temp}_{t-\Delta t-n}, v^{MSLP}_{t-\Delta t-1}, \ldots, v^{MSLP}_{t-\Delta t-n}, v^{wind}_{t-\Delta t-1}, \ldots, v^{wind}_{t-\Delta t-n}$, where the prediction horizons were set as 3-hour ahead,7-hour ahead and 12-hour ahead respectively. Thus, the initial input features can be considered as three univariate time series signals from three domains. In other words, the status in each previous time point is described with the wind speed, the temperature and the MSLP.

More than 260,000 records are selected for the series of experiments. In each experiment, 80% records are employed as training set and 20% records are employed for the testing. For different prediction horizons, the experiments were conducted via SVRs trained directed on the raw feature space with 108 dimensions (SVR), SVRs trained up in the feature space learned via the typical Autoencoder (AE-SVR), SVRs trained up in

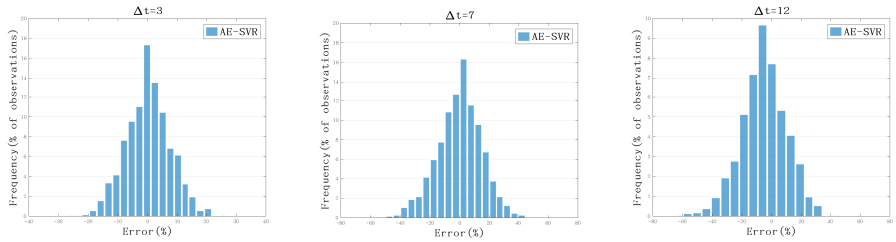
the feature space learned via split Autoencoder (SplitAE-SVR) and SVRs trained up in the feature space learned via the split-Autoencoder combined with canonical correlation term (CCA-SplitAE-SVR). As in section 5.2.4, the forecasting performance is evaluated via MSE and R^2 value. The detailed experimental results are given in the following section. Recently, LSTM is gradually more and more widely used in time-series applications. In this series of experiments, LSTM is also used for the univariate prediction and the results are compared with that of the other methods. To make the experiments more convincing, a distribution-oriented analysis of the forecast error is also given in this section, the skewness and kurtosis of the prediction results for each model were calculated and compared [105]. The skewness is the third moment of a distribution. It provides a measure of the asymmetry of the prediction error distribution. If the skewness is 0, then the distribution is symmetrical, if the skewness is negative then the distribution is left-skewed and if the skewness is positive,the distribution is right-skewed. The kurtosis is the fourth moment of a distribution, higher kurtosis means more of the variance is the result of infrequent extreme deviations, as opposed to frequent modestly sized deviations.

5.4.2 Experimental Results

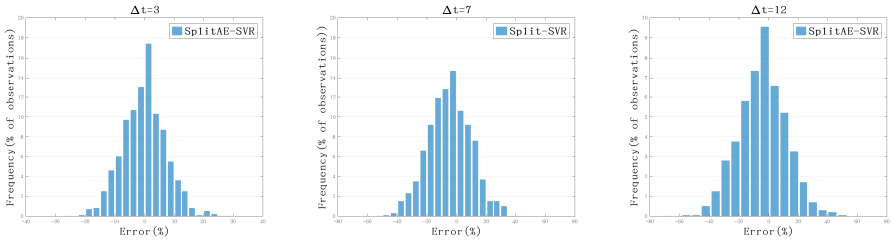
Table.5.4 gives the detailed experimental results. The prediction performances, include the MSEs and R^2 values of SVRs, AE-SVRs. SplitAE-SVRs, and CCA-SplitAE-SVRs trained up with multi-domain features are listed. Besides, the prediction results of SVRs and AE-SVRs trained up with univariate wind speed records are also included for comparison. According to the results, a predict model trained up with multi-domain features can provide higher prediction accuracy than it is trained up with univariate wind speed data. In detail, in the experiment, for 3-hour ahead prediction horizon, when us-



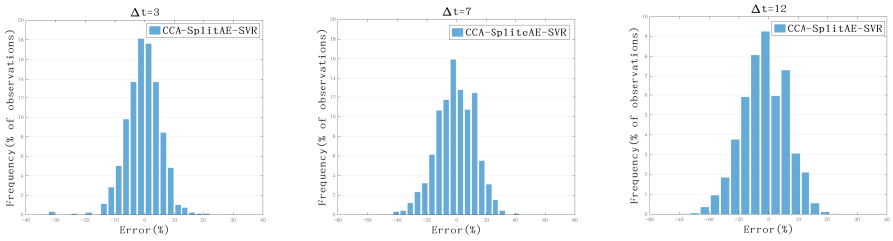
(a) Prediction error distribution histograms of SVRs for different prediction horizons



(b) Prediction error distribution histograms of AE-SVRs for different prediction horizons



(c) Prediction error distribution histograms of SplitAE-SVRs for different prediction horizons



(d) Prediction error distribution histograms of CCA-SplitAE-SVRs for different prediction horizons

Figure 5.7. Prediction error distribution histograms

ing multi-domain features, the SVR and AE-SVR obtained a 1% and 3% increase on R^2 value respectively, and the corresponding MSEs were also reduced. For 7-hour ahead prediction horizon, via SVR and AE-SVR, 0.5% and 2.1% gain on R^2 value were observed respectively when the models were trained up with multi-domain features. Such performance improvement was more significant for 12-hour ahead prediction horizon. For both of the two models, more than 3% R^2 increase were obtained. Meanwhile, the reduction of MSE were 0.3 and 0.12 respectively when the temperature and the pressure data were added as the input variables to train up these two models. These results demonstrate that the information from the temperature and pressure data is helpful to enhance the prediction accuracy, especially for prediction task with longer prediction horizon. It is therefore reasonable to improve the wind speed forecasting accuracy by using multi-domain data instead of only using univariate wind speed historical records to train up the prediction model. It is worth mentioning that the LSTM approach provides the best performance for the shortest time slot prediction task with the univariate model. This result shows the potential of the approach and we will try to find the feature fusion approach with LSTM in the future.

It has been demonstrated in Table.5.4 that relevant information from temperature data and pressure data can play a positive role in training up the wind speed forecasting model. The comparison among the performances of models trained up with the multiple domain data consequently shows the effects of utilizing the information from multiple domains by different models. According to the results, for all of the different prediction horizons, models with a representation layer all provided better performance than SVRs without any data fusion processing. Compared with AE-SVRs, SplitAE-SVRs and CCA-SplitAE-SVRs, SVRs trained up without a cross-domain data fusing operation provided 0.79, 0.71 and 0.64 R^2 for 3-hour, 7-hour and 12-hour ahead predic-

tion horizons respectively, which were at least 2%, 2% and 4% lower than other three models. These comparison results exhibit that a proper fusing representation of multi-domain features shall be benefit to the improvement of the forecasting performance of the adopted forecasting models.

Three feature representation models were given in this chapter for the feature fusion of the data from different domains, the AE model, the SplitAE model, and the proposed CCA based SplitAE model. According to the results in Table.5.4, for 3-hour and 12-hour ahead wind speed prediction horizons, SplitAE-SVR provided better forecasting performance than AE-SVR, while for 7-hour ahead prediction horizon, AE-SVR outperformed the SplitAE-SVR. However, the difference of R^2 values between AE-SVRs and SplitAE-SVRs for 3-hour, 7-hour and 12-hour ahead prediction horizons were 1.05%, 0.67% and 1.59%, which were all less than 1.6%; the difference of MSEs between AE-SVRs and SplitAE-SVRs for 3-hour, 7-hour and 12-hour prediction horizons were all less than 0.03. Such small differences on R^2 and MSE between these two models indicate that these two models provided comparable performances in the experiments. The difference of R^2 and MSE may be more attributed to the parameter configurations rather than the network architectures of these two models.

Compared with other models, the proposed CCA-SplitAE-SVR provided the best prediction performance. As illustrated in Table.5.4, for all the different horizons, CCA-SplitAE-SVR provided the highest R^2 values and the lowest MSE. The average enhancement of R^2 by using the SVRs trained up with the feature space represented via CCA based Split Autoencoder was close to 2.5% (Compared with SplitAE-SVR). Specifically, for 3-hour and 12-hour ahead prediction horizons, the enhancement of R^2 was larger than 3%. Correspondingly, the average reduction of MSE was over 0.9 (Compared with SplitAE-SVR). This demonstrated that the CCA-SplitAE-SVR is able to

improve the prediction accuracy of wind speed by the multi-domain feature fusing representation, and the CCA terms are helpful to utilize the useful relevant information more properly in temperature and pressure records for forecasting the wind speed.

As discussed above, the multi-domain feature representation models are also evaluated from the error distribution aspect. Fig.5.7 gives the distributions of prediction errors sampling from the prediction results for each model w.r.t. different prediction horizons. Table.5.5 gives the skewness and kurtosis of the prediction error distributions. From the skewness aspect, SVRs trained without feature fusion representation processing provided the lowest absolute value of skewness for all the different prediction horizons. For the other three models, almost all the absolute values of skewness were lower than 1.0, except the AE-SVR obtained a -1.31 skewness when $\Delta t = 12$. This fact demonstrates that the feature fusion representation may lead the forecasting model to be less symmetrical, but the loss of the asymmetry is acceptable. The positive influence of the proper feature fusing representation is demonstrated by the kurtosis of the error distributions. Significant enhancement of the kurtosis can be obtained after the feature representation operations, the CCA-SplitAE-SVRs provided the highest kurtosis for all the three prediction horizons, which were 2.27, 1.98, and 1.30 for 3-hour, 7-hour, and 12-hour ahead prediction horizons respectively. Moreover, for all the four models, a descending trend of kurtosis can be observed for longer prediction horizons. This shows that for longer prediction horizon, there are extreme deviations in the prediction results. The results about the skewness and the kurtosis can also be valid in Fig.5.7. Fig.5.7 visualizes the prediction errors. It is shown that the CCA-SplitAE-SVRs model has a sharper peak around the mode and longer tails with respect to a Gaussian distribution. Therefore, the proposed model leads to a sharper distribution of the errors and lower uncertainty.

5.4.3 Conclusive Discussion of the Experimental Results

According to the results, utilizing the relevant information from the temperature and pressure records to train up the forecasting model is helpful to improve the wind speed prediction accuracy. Apart from naively input all the data from different domains into the forecasting model, in this series of experiments, three ways of fusing the features from multiple domains are tested. The results show that a fusion representation of cross domain features can enhance the prediction accuracy in advance, especially the canonical correlation of the features from multiple domains are taken into consideration.

For the time series forecasting problem, the ARIMA models are most widely used in many applications [106]. What is more, for the feature representation of time series data, RNN has been widely employed in recent studies [128]. However, for wind speed forecasting, earlier studies have shown that ARIMA was outperformed by many neural models since the neural models perform better for simulating the nonlinear relationship among the variables [107]. More importantly, both of RNN and ARIMA may be not suitable for the fusing representation of data from different domains.

5.5 Conclusion

Focusing on the short-term weather forecasting application scenario, especially the wind speed forecasting problem, this chapter investigates the feature learning/representation issue for univariate time series data and multi-domain data records. Firstly, in the experiments for univariate data of temperature, pressure, and wind speed separately, Autoencoder model is demonstrated that can keep the time series relationship among the initial inputs after the feature representations. Consequently, based on the results in previous

experiments, several different versions of Autoencoder are adopted for the fusing representation of features from different domains. The experiments results show that the proposed CCA-SplitAE-SVRs can provide the best prediction performances.

The significance of this investigation is not only to enhance the prediction accuracy of short-term forecasting via machine learning models, but also to proposed a feasible model for the fusing representation of time series features from multiple domains. The CCA-Split Autoencoder model could also be a possible choice for other application scenarios. In this study, there is little discussion of RNN models which is widely considered more suitable for the feature representation of time series data since it may be difficult for RNN models of process features from multiple domains. In the future, we will pay more efforts on this issue.

Table 5.4. The wind speed prediction results by SVRs trained in feature spaces learned via different models

Prediction results for Δt =3		
	MSE	R^2
SVR(univariate)	1.5925	0.7811
SVR(cross domain features)	1.5099	0.7912
AE-SVR(univariate)	1.5132	0.7903
AE-SVR(cross domain features)	1.4517	0.8109
SplitAE-SVR	1.4221	0.8214
CCA-SplitAE-SVR	1.2554	0.8419
LSTM(univariate)	1.2531	0.8612
Prediction results for Δt =7		
	MSE	R^2
SVR(univariate)	2.0213	0.7141
SVR(cross domain features)	1.9053	0.7199
AE-SVR(univariate)	1.9008	0.7224
AE-SVR(cross domain features)	1.8278	0.7431
SplitAE-SVR	1.8005	0.7498
CCA-SplitAE-SVR	1.7012	0.7559
LSTM(univariate)	1.8012	0.7453
Prediction results for Δt =12		
	MSE	R^2
SVR(univariate)	4.4036	0.6124
SVR(cross domain features)	4.1025	0.6401
AE-SVR(univariate)	4.1041	0.6435
AE-SVR(cross domain features)	3.9912	0.6921
SplitAE-SVR	3.9873	0.7080
CCA-SplitAE-SVR	3.4125	0.7239
LSTM(univariate)	3.5468	0.7100

* Size/Acc.PCA.Red is the size/accuracy on the obtained subset via the PCA feature reduction pre-processing; Acc.PCA is the accuracy of the data set only with a feature reduction. Numbers in the parentheses are the numbers of features after PCA. e.g., HIGGS(14) means 14-feature HIGGS set.

Table 5.5. The comparison results for skewness and kurtosis of the prediction error distributions

	SVR		AE-SVR		Split-SVR		CCA-Split-AE SVR	
	S^k	k^b	S^k	k^b	S^k	k^b	S^k	k^b
$\Delta t = 3$	0.22	1.41	-0.71	1.81	-0.64	1.79	-0.92	2.27
$\Delta t = 7$	0.18	1.27	0.25	1.47	-0.81	1.42	-0.90	1.98
$\Delta t = 12$	-0.19	0.91	-1.31	1.19	0.45	1.21	-0.87	1.30

* Size/Acc.PCA.Red is the size/accuracy on the obtained subset via the PCA feature reduction pre-processing; Acc.PCA is the accuracy of the data set only with a feature reduction. Numbers in the parentheses are the numbers of features after PCA. e.g., HIGGS(14) means 14-feature HIGGS set.

Chapter 6

Feature Related Data Reduction

Feature representation cannot only affect the performance of employed computational models, but also related to other aspects of a certain machine learning applications, e.g., the number of necessary training samples. In this chapter, a novel data reduction approach that is highly related to the features of a date set is proposed.

Large scale data analysis is one of the major topics in Big Data related research [129]. Large scale data usually contains sufficient information that shall be helpful for data scientists to approximate the real distribution of the given data more closely [130, 131]. Processing massive data is a significantly more complex problem than processing those smaller data sets. It usually requires a lot of computational resources [129, 132, 133]. One of the strategies to lower the computational requirement is to calculate a representative of the raw data set with massive data records. Such operation is called data reduction or instance selection [134]. If the obtained subset can precisely represent the raw data set, researchers shall be able to approximate the real distribution of the given data set more closely with less computational resources by processing the obtained

subset instead of processing the raw data set.

Specific to machine learning research area, reasonably reduction of the size of the training sets is also meaningful in some application scenarios. A large scale data set enables complex machine learning models to be employed. For machine learning model with more parameters, a larger size of training set may lower the risk of overfitting [8]. However, in some application scenarios, the allowed computational recourse may be limited. For a machine learning model with a certain number of parameters, the output accuracy increases very quickly accompanying with the expansion of the training set at the beginning, while the required computation resource is relatively less at this stage; however, the increasing trend of the accuracy turns moderately when the training set is enlarged to a certain extent, and the demand of computational resources usually increases sharply then. This suggests that for a machine learning task on a certain data set with a very large size under limited computational recourse, it is possible to reduce the computational cost by properly shrinking the training set without much loss of the accuracy [133, 135].

Much effort has been invested on data reduction in earlier studies. Several previous approaches to calculate the subset are based on some forms of stratified random sampling [136, 137]. One of the main families of data reduction approaches are based on the k-Nearest Neighbors(k-NN) thoughts [138]. The main concept of this family of approaches to remove/keep a certain data point is according to the information from its k Nearest Neighboring points. Extended versions of this kind of approaches include Condensed Nearest Neighbor(CNN) [139], Selective Nearest Neighbor(SNN) [140], Edited Nearest Neighbor(ENN) [141], Instance-Based learning algorithm 2(IB2) and 3(IB3) [142] and the group of decremental reduction optimization procedures (DROP1-DROP5) [141]. These approaches differ in the way of the selection of initial samples,

the choice of K or the criterion to determine the neighborhood. Another big family of data reduction approaches is based on some clustering algorithms, such as k-means or fuzzy c-means algorithm [143]. For these approaches, the cluster centroids are selected as representatives of the prototypes of the raw data sets [134, 143, 144]. Clustering approach can also be considered a kind of sampling methods for data reduction. Since data reduction has been proved a NP-hard problem [145], local search heuristics and metaheuristics methods can also be employed to deal with this issue. In some previous studies, Tabu search, simulated annealing and genetic algorithms have been adopted for data reduction tasks [146–148], and memetic algorithms were proposed to select instances from the raw data set by combining the evolutionary algorithms and local search within the evolutionary cycle [149].

Reviewing the earlier data reduction approaches, different data reduction approaches focus on different aspects of the given data sets, meanwhile, all approaches have limitations. Randomly reduction is widely used and can provide acceptable results in some applications, however, the results may be unstable especially for imbalanced data set. Clustering-based approaches are trying to summarize the original distributions via prototypes, however, it is computational expensive and sensitive to the noise instances. Both of the random sampling approaches and clustering approaches are deterministic approaches that the size of the obtained subset should be pre-defined. This implies that prior knowledge about the given data set may be required. The family of k-NN based approaches are usually sensitive to the value of K, and actually the fundamental idea of k-NN based approaches is to select useful instances rather than to remove less necessary instances. Unfortunately, there is hardly any approach that can be well employed for all application scenarios, new thoughts for data reduction is worth to propose. Moreover, all the mentioned approaches are not specifically proposed for solving the data reduc-

tion on large scale data, the characteristics of large scale data set may not be utilized by some of the earlier proposed methods.

The proposed approach is to deal with the data reduction tasks specific to large scale data sets. Specifically, the proposed approach employs a Localized Generalization Error to evaluate how precisely the reduced data set can represent the raw large scale data set. A Q-neighborhood concept was introduced first, and the whole feature space can be divided into many Q-neighborhood grids according to the distribution of the samples w.r.t. each dimension. A bound of the gap between the generalization error from the original data set and that from the Q-neighborhoods is derived. According to the bound, selecting the representative samples as the obtained subset may represent the raw data set precisely. This work is an innovative extension of the research results in [150]. Compared with other approaches, the proposed methods need little prior knowledge for parameter setting, and is adaptive to the local data structure w.r.t. each dimension of the feature space. Supported by vigorous theoretical arguments, this method is significantly simple and intuitional while the performance is effective, and can be easily parallel possessed for acceleration. It is a feasible strategy for the instance reduction of some large data sets. The proposed method is tested on several large scale data sets from UCI. In the experiments, a subset with a much smaller volume ($10\% \sim 40\%$ of the raw data) can be acquired, meanwhile, learning models can obtain a comparable accuracy (1% accuracy decay) on the subset.

The remaining part of this chapter is organized as follows. Section 6.1 gives some preliminaries of the proposed model, including a mathematical definition of a large scale data set and a brief review of the localized generalization error. The definition of Q-neighborhood grid and the detailed algorithm are presented in Section 6.2. Section 6.3 contains the experimental results. Finally in the conclusion section, the main contents

are summarized, the limitations are analyzed and the future works are suggested.

6.1 Preliminaries

In this section, several important concepts are discussed, including the mathematical description of the term "large scale data", and a brief review of the localized generalization error.

6.1.1 Large scale data

In a formal statistical learning framework, based on the *i.i.d. assumption*, set the real distribution as $\mathcal{D}$, given the hypothesis class $\mathcal{H}$, $\delta \in (0,1)$ and $\epsilon > 0$, let m be the integer that satisfies:

$$m \geq \frac{log(|\mathcal{H}|/\delta)}{\epsilon}, \tag{6.1}$$

then for any labeling function f, with the probability of at least $1-\epsilon$ over the data set S including m samples, respecting every *Empirical Risk Minimization* (ERM) [7, 151] hypothesis $h_S \in \mathcal{H}$, it holds $R_{(\mathcal{D},f)}(h_S) \leq \epsilon$, where R is for the predefined error such as least square [131].

Consequently, a formal learning model, the *Probably Approximately Correct* (PAC) learnability of the hypothesis of $\mathcal{D}$ can be described as:

A hypothesis class $\mathcal{H}$ is PAC learnable if there is a function to determine a $m_{\mathcal{H}} \in \mathbb{N}$ and a learning algorithm can satisfy that, for every $\delta, \epsilon \in (0,1)$, for every distribution $\mathcal{D}$ over $\mathcal{X}$ and label function f, when running the learning algorithm on $m \geq m_{\mathcal{H}}$ i.i.d samples generated by $\mathcal{D}$ and labeled by f, the algorithm can return a hypothesis h with

probability at least $1-\eta$, $R_{(\mathcal{D},f)}(h_S) \leq \epsilon$ [152].

According to Eq.6.1, it is easily to obtain a corollary that every finite hypothesis class is PAC learnable with the sample complexity $m_{\mathcal{H}}(\delta,\epsilon)$ that has the lower bound $\frac{log(|\mathcal{H}|/\delta)}{\epsilon}$.

Furthermore, a set S is called ϵ-representative w.r.t. a real distribution $\mathcal{D}$ if :

$$\forall h \in \mathcal{H}, |R_S(h) - R_{\mathcal{D}}(h)| \leq \epsilon \tag{6.2}$$

It can be deduced that, for finite hypothesis class $\mathcal{H}$, if S is a sample set with $m \geq m_{\mathcal{H}}(\delta,\epsilon)$ of examples drawn from i.i.d according to $\mathcal{D}$, then, with the probability of at least $1-\delta$, S is ϵ-representative of $\mathcal{D}$.

The above discussion reveals the fact that for a certain finite hypothesis class $\mathcal{H}$, a larger data set, provides smaller difference between the error on the training set S and the error on real distribution $\mathcal{D}$ over $\mathcal{X}$. Thus, in this chapter, the term large scale indicates that the quantity of training samples satisfies:

$$m_S \gg \frac{log(|\mathcal{H}|/\delta)}{\epsilon} \tag{6.3}$$

where $|\mathcal{H}|$ is considered relating to the number of features and the parameters of the employed learning model. Under the condition of Eq.6.3, S can be considered as a closely approximate representation of $\mathcal{X}$, and $R_S(h) \simeq R_{\mathcal{D}}(h)$.

6.1.2 Brief about localized generalization error

The main idea of this study is trying to calculate a subset of the original training set, and make sure the subset can approximately represent the original data set. To evaluate the precise degree of such representation, the localized generalization error is adopted as the measure.

The localized generalization error R_{SM} was originally proposed in [150] . It bounds from the generalization error for unseen samples within a predefined neighborhood of the training samples using stochastic sensitivity measure. Given a training set S containing m samples according to $\mathcal{D}$ over $\mathcal{X}$, $S = \{\mathrm{x}_b, F_{\mathcal{D}}(\mathrm{x}_b)\}_{b=1}^{m}$, f_θ is a classifier. The generalization error is defined as:

$$R = \int_{\mathcal{X}\setminus S} (f_\theta(\mathrm{x}) - F_{\mathcal{D}}(\mathrm{x}))^2 \, p(x) d\mathrm{x} \tag{6.4}$$

For every sample $\mathrm{x}_b \in S$, one finds a set of samples x which fulfills $0 < |\Delta x_i| < Q$, $\forall i = 1, \ldots, n$, where n is the number of the dimensions, and $\Delta\mathrm{x} = (\Delta x_1, \ldots, \Delta x_n)^T = \mathrm{x} - \mathrm{x}_b$. The Q-neighborhood of a training sample x_b thus can be given as:

$$H_Q(\mathrm{x}_b) = \{\mathrm{x} | \mathrm{x} = \mathrm{x}_b + \Delta\mathrm{x}; |\Delta x_i| \leq Q, \forall i = 1, \ldots, n\} \tag{6.5}$$

Based on the Eq.6.4, for $0 \leq Q_1 \leq \cdots \leq Q_k \leq \infty$, the following relationship holds:

$$\mathrm{x}_b \subseteq H_{Q_1}(\mathrm{x}_b) \subseteq \cdots \subseteq H_{Q_k}(\mathrm{x}_b) \subseteq \mathcal{X} \tag{6.6}$$

The shape of the Q-neighborhood is chosen to be a hypercube for ease of com-

putation. Figure.6.1 is a simple illustration of the Q-neighborhoods in a small two-dimensional training set.

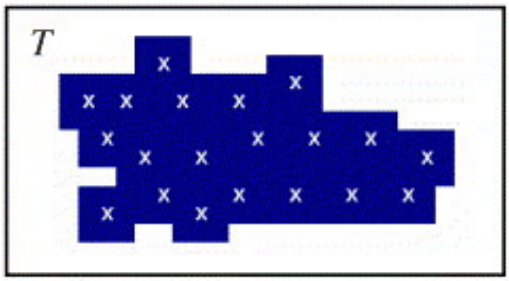


Figure 6.1. Illustration of Q-neighborhoods of 20 training samples. The xs are training samples and the Q-neighborhoods of each training sample is a hypercube, and T denotes the entire space of $\mathcal{X}$.

Defining $err_\theta(\mathrm{x}_b) = f_\theta(\mathrm{x}_b) - F_\theta(\mathrm{x}_b)$, the empirical risk can be denoted as $R_{emp} = \frac{1}{m}\sum_{b=1}^{m}(err_\theta(\mathrm{x}_b))^2$. Using T_Q to denote the union of Q-neighborhoods which is the shaded area in Figure.6.1, defining $\Delta(y) = f_\theta(\mathrm{x}) - f_\theta(\mathrm{x}_b)$, there are $E_{T_Q}((\Delta y)^2) = \frac{1}{m}\sum_{b=1}^{m}\int_{T_Q(\mathrm{x}_b)}\left((\Delta \mathrm{y})^2\right)\frac{1}{(2Q)^n}d\mathrm{x}$, and $\epsilon = B\sqrt{\ln\eta/(-2m)}$, where A and B be the difference between the maximum and minimum values of the target outputs, the maximum possible value of the MSE, then the generalization error for the union of the Q-neighborhoods and its upper bound, by the Hoeffding's inequality [153], with probability of $1-\eta$, can be given in Eq.6.7 [150, 154].

$$
\begin{aligned}
R_{SM}(Q) &= \int_{T_Q} (f_\theta(\mathrm{x}) - F_{\mathcal{D}}(\mathrm{x}))^2 \, p(x) d\mathrm{x} \\
&\leq \frac{1}{m} \sum_{b=1}^{m} \int_{T_Q(\mathrm{x}_b)} (f_\theta(\mathrm{x}) - F_{\mathcal{D}}(\mathrm{x}))^2 \frac{1}{(2Q)^n} d\mathrm{x} + \varepsilon \\
&= \frac{1}{m} \sum_{b=1}^{m} \int_{T_Q(\mathrm{x}_b)} (f_\theta(\mathrm{x}) - f_\theta(\mathrm{x}_b) + f_\theta(\mathrm{x}_b) - F_{\mathcal{D}}(\mathrm{x}_b) + F_{\mathcal{D}}(\mathrm{x}_b) - F_{\mathcal{D}}(\mathrm{x}))^2 \frac{1}{(2Q)^n} d\mathrm{x} + \varepsilon \\
&\leq \frac{1}{m} \sum_{b=1}^{m} \int_{T_Q(\mathrm{x}_b)} \left((\Delta y)^2 \right) \frac{1}{(2Q)^n} d\mathrm{x} + \frac{1}{m} \sum_{b=1}^{m} \int_{T_Q(\mathrm{x}_b)} \left((err_\theta(\mathrm{x}_b))^2 \right) \frac{1}{(2Q)^n} d\mathrm{x} \\
&\quad + \frac{1}{m} \sum_{b=1}^{m} \int_{T_Q(\mathrm{x}_b)} \left((F_{\mathcal{D}}(\mathrm{x}_b) - F_{\mathcal{D}}(\mathrm{x}))^2 \right) \frac{1}{(2Q)^n} d\mathrm{x} \\
&\quad + 2 \sqrt{\left(\frac{1}{m} \sum_{b=1}^{m} \int_{T_Q(\mathrm{x}_b)} \left((\Delta y)^2 \right) \frac{1}{(2Q)^n} d\mathrm{x} \right) \left(\frac{1}{m} \sum_{b=1}^{m} \int_{T_Q(\mathrm{x}_b)} \left((err_\theta(\mathrm{x}_b))^2 \right) \frac{1}{(2Q)^n} d\mathrm{x} \right)} \\
&\quad + 2 \sqrt{\left(\frac{1}{m} \sum_{b=1}^{m} \int_{T_Q(\mathrm{x}_b)} \left((err_\theta(\mathrm{x}_b))^2 \right) \frac{1}{(2Q)^n} d\mathrm{x} \right) \left(\frac{1}{m} \sum_{b=1}^{m} \int_{T_Q(\mathrm{x}_b)} \left((F_{\mathcal{D}}(\mathrm{x}_b) - F_{\mathcal{D}}(\mathrm{x}))^2 \right) \frac{1}{(2Q)^n} d\mathrm{x} \right)} \\
&\quad + 2 \sqrt{\left(\frac{1}{m} \sum_{b=1}^{m} \int_{T_Q(\mathrm{x}_b)} \left((\Delta y)^2 \right) \frac{1}{(2Q)^n} d\mathrm{x} \right) \left(\frac{1}{m} \sum_{b=1}^{m} \int_{T_Q(\mathrm{x}_b)} \left((F_{\mathcal{D}}(\mathrm{x}_b) - F_{\mathcal{D}}(\mathrm{x}))^2 \right) \frac{1}{(2Q)^n} \mathrm{x} \right)} + \varepsilon \\
&\leq \left(\sqrt{R_{emp}} + \sqrt{E_{T_Q}\left((\Delta y)^2 \right)} + A \right)^2 + \varepsilon
\end{aligned}
\tag{6.7}
$$

Both A and ϵ are constants for a given training data set when an upper bound of the classifier output values is preselected. The result of Eq.6.7 is an upper bound for the generalization error of the trained classifier for unseen samples within the union of the Q-neighborhoods. This error bound gathers on the statistical characteristics of training data set such as its mean and variance, and grows slowly with the increase of the number of parameters.

The specific value of the term $E_{T_Q}\left((\Delta y)^2\right)$ in Eq.6.7 depends on the form of the

learning model employed. However, the generalization error $R_{SM}(Q)$ and its bound could be defined for any classifier trained with MSE. Examples include feedforward neural networks like MLPNN, SVM, and Recurrent neural networks such as Hopfield networks. Classifiers such as rule-based system and decision tree may not be able to make use of this concept.

In summary, section 6.1.2 shows a fact that if a certain training set is given, like the 20 training samples in Figure.6.1, for other i.i.d data points belonging to the Q-neighborhood (in the shaded area), the upper bound of the generalization error from these data is shown in Eq.6.7. In the next section, the sample reduction strategy based on the inverse process of this fact will be introduced.

6.2 The Q-neighborhoods based Data Reduction

In this section, the proposed Q-neighborhood based Data reduction will be introduced. In detail, the upper bound difference of the localized generalization error between the Q-neighborhoods and the samples can be deduced from Eq.6.7, this fact provides a chance to make sure the classification accuracy drops little after the reduction. Based on this fact, an inverse condition of section 6.1.2 can be considered. The proposed approach is then given in detail.

6.2.1 The inverse condition

According to the discussion in section 6.1.2, the upper bound of the localized generalization error of the union of the Q-neighborhoods contains four terms, the training error of all x_bs denoted by R_{emp}, the $E_{T_Q}\left((\Delta y)^2\right)$ which is determined by Q and the

constants A and ϵ. A can be preselected and ϵ will be very small if the training data size m is very large. So there is:

$$\begin{aligned} |R_{SM}(Q) - R_{emp}| \leq & 2\sqrt{R_{emp}}(\sqrt{E_{T_Q}\left((\Delta y)^2\right)} + A) \\ & + (\sqrt{E_{T_Q}\left((\Delta y)^2\right)} + A)^2 + \epsilon \end{aligned} \tag{6.8}$$

$R_{SM}(Q)$ is the generalization error for the shaded area in Figure.6.1, and R_{emp} is the training error for a limited number of training samples x_bs. It is reasonable to assume $\exists f_\theta$, $R_{emp}(f_\theta) \leq \epsilon_1$, where ϵ_1 is a small positive number. Thus, $\exists f_\theta$, which makes:

$$\begin{aligned} |R_{SM}(Q, f_\theta) - R_{emp}(f_\theta)| \leq & 2\sqrt{\epsilon_1}(\sqrt{E_{T_Q}\left((\Delta y)^2\right)} + A) \\ & + (\sqrt{E_{T_Q}\left((\Delta y)^2\right)} + A)^2 + \epsilon \end{aligned} \tag{6.9}$$

Moreover, according to the PAC learnability in section 6.1.1, it can be deduced that, with the probability of $1 - \delta$, there is

$$|R_{SM}(Q) - R_{SM}(Q, f_\theta)| \leq \epsilon_2 \tag{6.10}$$

if the number of samples $m \leq \frac{log(|\mathcal{H}|/\delta)}{\epsilon_2}$. Therefore, for a preselected η, ϵ_2 is actually determined by m. Combine Eq. 6.9 and Eq. 6.10 together:

$$
\begin{aligned}
|R_{SM}(Q) - R_{emp}(f_\theta)| \leq 2\sqrt{\epsilon_1}(\sqrt{E_{T_Q}\left((\Delta y)^2\right)} + A) \\
+ (\sqrt{E_{T_Q}\left((\Delta y)^2\right)} + A)^2 + \epsilon + \epsilon_2
\end{aligned}
\tag{6.11}
$$

The above discussion presents an upper bound of the difference between the generalization error from the training samples x_bs and the union of Q-neighborhoods of x_bs (the shaded area in Figure 6.1). **Based on this fact, an inverse condition of Eq.6.7 shall be assumed. Assuming that the shaded area does not indicate the unseen samples, conversely, it represents the input data collections. Moreover, if the shaded area is divided into many grids as Q-neighborhoods, and the** xs **in Figure 6.1 thus can be deemed as selected representative samples from the grids.** In this scenario, the upper bound of the gap between the error from the shaded area and that from the samples is still applicable. More importantly, since the volume of a grid is determined by the value of Q, the total number of the grids therefore is depending on Q. In advance, the number of the selected samples, m, is also determined by Q. As discussed in above, ϵ_2 is therefore actually determined by m. Putting all the above relations together, ϵ_2 is a function of Q.

Checking the right side of Eq.6.11, ϵ and ϵ_1 are small positive constants, A is a preselected constant. For the other terms, $E_{T_Q}\left((\Delta y)\right)^2$ is a function of Q, although the function form may vary if different models are employed [150], and ϵ_2 is also a function of Q.

In summary, in the proposed methods, the raw input samples are seen as the shaded area in Figure 6.1, consequently the space can be cut into many grids based on Q-neighborhood, and representative samples are chosen from each grid. The difference of

the generalization errors between training on the raw input data set and on the selected samples has an upper bound determined by Q. This idea can be considered as an inverse process in Eq.6.7.

6.2.2 The proposed data reduction approach

According to the previous discussion, the goal of the proposed approach is to cut the input data space into many Q-neighborhood grids, and select representative samples from the grids, then by using the selected samples as training set instead of using the whole input data set, the employed classifier can obtain a comparable classification accuracy.

Assuming there are totally m training samples in the initial given data sets S_{ini}, each training sample $\mathrm{x}_i \in \mathbb{R}^n$, $\mathrm{x} = \{x_1, \ldots, x_n\}$, $i = 1, \ldots, m$ and with a corresponding label $y_i, i = 1, \ldots, m$, our strategy is to choose a certain integer N, for the kth feature, $k = 1, \ldots, n$, depending on the range of the data values corresponding to this feature, $[\min(x_k), \max(x_k)]$, the value range of this feature can be divided into N equal intervals, and the length of each interval is the value of Q for the ith feature. Since the ranges of the data values in different features can vary, the values of Q for different features is not the same. However, this can be easily solved by scaling the features, which makes all values in the data sets be mapped into a same-length range, such as $[-1, 1]$. Thus the whole input space can be divided into N^n grids, each grid is a hypercube that satisfies the definition of Q neighborhood in Eq.6.5 w.r.t. the centroid of the grid.

The following step is to select representative samples for the grids. In this step, several different conditions should be considered:

(1) if there is only one sample in a certain grid, this sample contains all the infor-

mation of this grid, thus this sample should be put into the selected subset to represent the information of this grid;

(2) if there are more than 1 sample in a certain grid, and all the samples have the same class label, the strategy is to use the mean value of these samples to represent the information in this grid, and keep the class label. Denote the representative sample as x', From Eq.6.5, all the samples in this grid belong to a Q'-neighborhood of the mean points, where $Q' \leq Q$. According to Eq.6.11, the difference between $R_{SM}(Q')$ and the $R_{emp}(x', f_\theta)$ is upper bounded in $2\sqrt{\epsilon_1}(\sqrt{E_{H_{Q'}}\left((\Delta y)^2\right)}+A)+(\sqrt{E_{H_{Q'}}\left((\Delta y)^2\right)}+A)^2+\epsilon+\epsilon_2$;

(3) if there are more than 1 sample labeled differently in a certain grid, while the quantities of differently labeled samples are not equal, the strategy is to obtain the majority information in this grid. Firstly, a threshold $\varrho \in (0.5, 1]$ is given to control the quantity of the information discarded. Set the total number of samples in this grid as m_g, and the number of samples labeled differently as $m_{g1}, m_{g2}, \ldots$ corresponding to $y_1, y_2, \ldots$, where $m_{g1} \geq m_{g2} \geq \ldots$. In the beginning, all samples labeled y_1 are selected. If $m_{g1}/m_g > \varrho$, these samples are used to represent this grid; otherwise, samples labeled y_2 are consequently added into the reduced set, until more than $p \times m_g$ samples are added. Such operations ensure that samples in this grid can be classified into the selected classes with more than ϱ probability. From information theory aspect, the upper bound of the information loss in this grid is $-(1-\varrho)\log(1-\varrho)$;

(4) if there are equal number of sample labeled differently in a certain grid, all the samples should be reserved to keep this part of information in the raw data set.

Some comments to the proposed approach are given as follows.

(1) The proposed approach is focusing on the data reduction issue of large scale

data set. The large scale of the input data set is necessary, and the characteristics of large scale data is utilized by the proposed approach. Take the condition in Figure.6.1 as example, Eq.6.11 shows that the differences of the localized generalization error between the shaded area ($R_{SM}(Q)$) and the x_bs ($R_{SM}(Q)$ and R_{emp}) are upper bounded, therefore the x_bs can be employed as representative instances for the dark area. In real applications, it is expected to approximate the real distribution $\mathcal{D}$ over $\mathcal{X}$ (the entire space T). When the quality of the data set is not sufficient, the differences between the entire space T and the Dark area should not be ignored. In contrast, in large scale data learning scenario, the gap between the dark area and T shall be very small. As shown in section 6.1.1, when m_{ini} is large enough, S_{ini} (dark area) could be employed to approximate $\mathcal{D}$ (over $\mathcal{X}$) with an extremely small error, then a proper selected subset (x_bs) from S_{ini} may be adopted to closely approximate $\mathcal{D}$. What is more, since all the samples are i.i.d, it can be reasonably assumed that the density of samples in a certain grid is higher when the m_{ini} is larger. This may lower the bias during representative sample selection for each grid.

(2) The number of samples in the reduced data set cannot be predefined. One of the advantages of the proposed approach is the adoptive processing on the given data set. Controlled by the localized generalization error, the proposed approach will not forcibly remove some useful samples to reduce the original data set to a certain size.

(3) The size of the obtained subset highly depends on the choice of N. The value of Q is based on N. A smaller value of N means bigger volume of each grid, and therefore with higher probability that the initial data can be reduced to a smaller one through the proposed approach. Conversely, if $N \to \infty$, the reduced set will be equal to the original set. Moreover, since the density of the input samples is not uniform, the relationship between the size of the reduced set and N is monotonic but not linear.

(4) In close examination, because the selected samples are not necessarily locating on the centroid position, these samples don't have a strict Q-neighborhood with other areas in the corresponding grids. However, since most of these samples are selected from the originally data set, using these samples shall be more reliable than using the grid centroids.

(5) Compared with k-NN based approaches and clustering methods, the proposed approach is easily quickened through parallel processing.

6.3 Experiments and Extension Approach with Weights Assignment

In this section, a series of experiments were conducted to evaluate the proposed approach. Moreover, several extension approaches are also given to obtain a better performance.

6.3.1 Experiments to test the proposed approach

Several real large scales data sets (the ratio of number of samples to number of features $\geq 10^4$) were selected from UCI as benchmarking data. For each data set, about 10% samples were randomly selected for testing, the remaining 90% samples were used as initial training set. By running the proposed Q-neighborhood sample reduction algorithm, a subset of the initial training set was given. Classifiers were trained with the initial training set and the subset respectively, and tested with the same testing set. The difference of the testing accuracies is the key criterion to verify the proposed approach. To make the results more convincing, an artificial data set was also applied for the test-

Table 6.1. The detailed data sets information

Data Set	# Samples	# Features	#Classes
HIGGS	11,000,000	28	5
Skin Seg (SS)	245,057	4	2
SUSY	5,000 ,000	14	2
Statlog	58,000	9	5
Artificial Data	9,000,000	30	10

ing. Table.6.1 shows the detailed information about the data sets.

The reduction performance of the proposed results are mainly evaluated from two aspects. The size of the obtained subset shows the reduction capacity of the proposed approach on the processed data set and the accuracy decay shows the drop of the accuracy when using the obtained subset as training set instead of using the original data set. The obtained subset is expected with a smaller size and classifier trained on the obtained subset is expected with a smaller accuracy decay. The performance of the proposed approach was tested first on the 4 benchmarking data sets. In the experiment, for each benchmarking data set, firstly, subsets were calculated via the proposed approach. the classification accuracy of the models was tested next.

The random sampling approach was employed as baseline for comparison. A randomly-selected subsets with the same size of the calculated subset was also employed as training set. To reduce the effect from parameters of the classifier on the accuracy, three-layer Nerual Networks were employed as the classifiers in the experiment. The three-layer Nerual Network is reported can be adopted and offer satisfy performance for most data sets. It is a vary universal classification model and little parameters need to be pre-defined if the number of hidden neurons is fixed. Figure 6.2 shows the reduction rate (ratio of the reduced data set size to the original data set size) of the data sets on different values of N. The curve in Figure.6.2 illustrates the change of the reduction

[1] For large scale data set, Support Vector Machines (SVMs) usually provide similar performance as NNs. In principles, these two methods actually reach the same goal by different routes. However, SVMs need more parameters to adjust.

Table 6.2. Experimental results

Data Set	Size. ini	Size. Red	Acc.ini	Acc.Red	Acc. Rand	Reduction Ratio	Acc. Diff.Red.	Acc. Diff.Rand.	Value of N
	58,000	8,010	90.48(3.18)	90.24(2.94)	85.29(2.53)	0.1381	-0.24	-5.19	450
Statlog	58,000	6,543	90.48(3.18)	87.48(3.25)	84.60(3.25)	0.1128	-3.00	-5.88	300
	58,000	5,055	90.48(3.18)	84.62(2.07)	84.98(1.19)	0.0872	-5.86	-5.50	150
	245,057	51,433	95.60(0.28)	97.27(1.21)	89.88(5.31)	0.2099	+1.67	-5.72	180
SS	245,057	47,729	95.60(0.28)	93.32(1.59)	90.97(4.45)	0.1948	-2.28	-4.63	120
	245,057	37,725	95.60(0.28)	91.12(2.31)	80.66(7.13)	0.1536	-4.48	-14.94	40
	5,000 ,000	3,072,458	77.59(2.51)	77.60(2.90)	76.86(3.15)	0.6145	+0.01	-0.73	10
SUSY	5,000 ,000	1,547,084	77.59(2.51)	76.90(2.12)	75.88(3.23)	0.3094	-0.69	-1.71	8
	5,000 ,000	472,503	77.59(2.51)	73.91(1.91)	68.76(1.70)	0.0947	-3.68	-8.83	6
	11,000,000	10,720,516	75.80(1.32)	75.46(0.47)	75.30(0.78)	0.9746	-0.34	-0.50	10
HIGGS	11,000,000	9,713,357	75.80(1.32)	75.10(1.15)	75.25(1.39)	0.8830	-0.70	-0.55	8
	11,000,000	8,159,691	75.80(1.32)	72.12(3.70)	60.21(2.12)	0.7418	-5.68	-15.59	6

* Size.ini is the size of the initial data set, Size. red is the size of the data set after reduction; Acc.ini/Red/Rand is the classification accuracy of the model trained on the initial data set/ date set reduced by the proposed approach/data set reduced by randomly selection; Reduction Ratio is the Ratio between Size.Red and Size.ini; Acc.Diff.Red./Rand. is the accuracy decay on the subset obtained by the proposed approach/data set reduced by randomly selection where a positive value indicates an accuracy improvement and a negative value indicates an accuracy drop.

capacity with different N. The numbers on the curve represent the size of the reduced set.

From Figure.6.2 and Table.6.2, for all the four data sets, despite some small volatility, the size of the reduced data sets obviously decreased while the parameter N was set with a larger value. Moreover, the reducing rate is also affected by the structure of the processed data set. For the Statlog and SS data sets, even N was set to a 10^2 larger level, our approach can still remove more than 80% samples. For SUSY, only 40% of samples can be removed with $N = 10$. The worst condition is for the HIGGS data set, less than 10% samples removed after the reduction with $N = 10$. To explain the results, for a fixed density n dimensional data set, there are totally N^n grids in feature space. When N is fixed, the number of grids will exponentially increase accompany with the increase of n. For higher dimensional data the reduction capacity of this approach may not be that significant.

Figure.6.3 shows the performances on accuracies with different N. In each subfigure, the red horizontal line is the classification accuracy of the classifier trained on

the raw data, the blue solid curve is the accuracy of the classifier trained on the reduced set, and green dash curve is the average accuracy of the classifier trained on the same-size randomly selected subset. The classification accuracy decays after the reduction are also indicated in each sub-figure. More detailed results of accuracy are given in Table.6.2.

Together with Figure 6.2, the results in Figure 6.3 show that the classifier can provide a comparable accuracy with a much smaller training set obtained by the proposed approach. By ignoring the most extremely cases such as $N = 2$, in most cases, the accuracy decay was lower than 4% by using less than 20% of the total training samples. Also, the accuracy increased accompany with the larger value of N was set. The content in Table.6.2 gives the results of the model with 3 different values of N selected for each data set, including the reduction where the highest accuracy was obtained. For SS, SUSY and Statlog, the error difference is controlled within ±2%, while the data set was greatly reduced ($39\% \sim 80\%$), and the training time was also significantly reduced ($49\% \sim 74\%$) in the best case.

Some points should be emphasized in this experiment. (1) For a certain data sets, a small accuracy improvement (1.67% on SS and 0.1% for SUSY) can be obtained when training the model on the reduced sets, this may be explained by the fact that our approach can remove noises with a proper value of N; (2) The Staglog data set is an imbalanced data set with multiple labels, in such scenario, the randomly selected subset sometimes may omit the information of the smaller classes. The comparison between the two approaches on this data set verifies the reliability of the proposed model; (3) k-NN based approaches were not compared in the experiment. First, the performance of k-NN is quite sensitive to the parameter k; second, k-NN needs huge computational resources when processing a large number of samples. Current results are sufficient to

suggest that the proposed approach is a feasible choice for data reduction on a large scale data set.

6.3.2 Extension 1: Adding PCA on the proposed approach and experiment

The first experiment shows that for some large scale data sets, by employing the proposed approach, a subset for representing the original data set can be obtained to train the classifiers without a significantly drop of the classification accuracy. However, observing the experimental results, the reduction capacity of the proposed approach varies on different data set, e.g., the reduction effect for the HIGGS data set was quite weak. This kind of results may be due to the larger number of features in the data set, for higher dimensional feature space, there will be more Q-neighborhood grids obtained. Consequently for a certain sample, with higher possibility, it will be reserved in the obtained subset.

A reasonable deduction of the previous experiment and analysis is that for a certain data set, the reduction capacity of the proposed approach may be enhanced by giving the data set a proper feature reduction before the sample reduction operation. In the following experiment, a feature reduction operation was given to the processed data set first, i.e., the initial data set was mapped into a lower dimensional feature space via PCA, and then tested the proposed model ($N = 10$) on the HIGGS data set with less features. To make the results more convincing, the SUSY data and an artificial data set (ART) were also employed in this experiment. The results are shown in Table.6.3.

As shown in Table.6.3, for 28-feature data set HIGGS, a 14-feature and 9-feature set were obtained via PCA respectively. In previous experiment, for HIGGS, the capac-

Table 6.3. The results after PCA

	HIGGS with 14 features	HIGGS with 9 features	SUSY with 10 features	ART with 15 features
Size.ini	11,000,000	11,000,000	5,000,000	9,000,000
Size.PCA.Red	2,273,439	1,014,370	271,384	998,342
Red.Ratio	0.21	0.09	0.05	0.11
Acc.ini	75.80	75.80	77.60	81.52
Acc.PCA	76.01	73.14	76.20	80.43
Acc.PCA.Red	72.92	70.07	73.38	79.35

* Size.PCA.Red is the size the obtained subset via the PCA feature reduction pre-processing; Acc.PCA is the classification accuracy trained on the initial data set only with PCA feature reduction, Acc.PCA.Red is the classification accuracy of the classifier trained on the subset via the PCA feature reduction pre-processing.

ity of the proposed approach is weak, only 2% was removed when $N = 10$. Now for the 14-feature set, About 80% samples are removed, while the accuracy decay was less than 2.7%. However, for the 9-feature set, more samples are removed, but the accuracy drop was larger than 5% since more information was excluded. The condition is similar for other data sets. The results showed that a proper feature reduction processing can obviously enhance the reduction effects of the proposed approach.

6.3.3 Extension 2: Weighted Q-neighborhood sample reduction and experiment

According to the definition of Q-neighborhood in Eq.6.5, for a certain sample x, its Q-neighborhood is a hypercube which takes x as center and select a Q length range on each dimension. The results of the previous experiment also demonstrates that the proposed approach is highly related to the distribution of the samples in each dimension of the feature space. In previous sections, all the dimensions of the feature space are considered equally important. However, according to Eq.6.5, Eq.6.11 and the results of the previous experiments, when the parameter N is set a smaller value (means each Q-neighborhood grid is bigger), the reduction performance might be higher but more classification accuracy will be scarified. Therefore, if the value of N for each dimension

varies according to the importance of the corresponding dimension, i.e., for a more important dimension, a larger N is assigned to keep more information w.r.t. this dimension while for a less important dimension a smaller N can be given, the proposed model may remove more samples with less accuracy decay. In this section, as a further extension of the proposed approach, a different value of N is assigned on each dimension to obtain the weighted Q-neighborhood grids, the performance of the weighted Q-neighborhood sample reduction approach is also evaluated with experiment.

The main strategy of the weighted Q-neighborhood sample reduction is to decide whether this feature should be assigned a parameter N with a bigger value or a smaller value by considering the importance of a certain feature. Currently, there are mainly two approaches to decide the importance of a dimension:

1. **PCA and Eigenvalues**

   PCA has been used for feature reduction in previous experiment, the eigenvalues obtained in PCA indicated how large the variability of the data distributed in the direction of the corresponding eigenvector. This can be used to measure the importance of each dimension in the feature space of the principal components. Therefore, the eigenvalues can be employed to determine whether the corresponding feature should be given a parameter N with a bigger value or a smaller value.

2. **Mutual Information** PCA focuses on the information of each dimension [17], Mutual Information considers the relationship between each feature and the class labels. For a certain feature x_i, its Mutual Information with the label y is defined as $I(x_i, y) = H(x_i) - H(y|x_i)$. Therefore, when using Mutual Information to measure the importance of each dimension, the effect of the class labels is also taken into consideration. Since the employed data set, the value of the attributes

Table 6.4. The results of the weighted Q-neighborhood based data reduction

	HIGGS	SUSY	ART
Size.ini	11,000,000	5,000,000	9,000,000
Size.PAC.weighted	828,426	63,472	952,412
Acc.PAC.weighted	73.44	69.13	79.81
Size.MI.weighted	923,076	230,676	937,353
Acc.MI.weighted	70.80	74.36	79.35

* Size.PCA/MI.weighted is the size the obtained subset via the PCA/Mutual Information based weighted Q-neighborhood; Acc.PCA/MI. is the classification accuracy of the model trained on the corresponding obtained subset

is continuous while the value of the label vector is categorical, kernel density estimation is employed to estimate the probability density function [155].

In the experiment, for each data set, both of the PCA and Mutual Information are employed to determine the importance of each dimension, and then different N values are assigned to each dimension according to the dimension importance. Table.6.4 gives the results of the experiment.

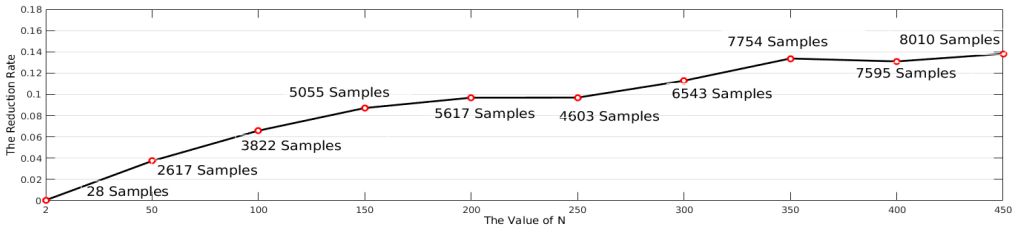
The data sets employed in section 6.3.2 were also employed in this experiment, and the results are various on different data sets. For the HIGGS data set, the PCA weighted processing can improve the reduction capacity in advance, the obtained subset was 19% smaller than the one in section 6.3.2, while there is even a small improvement on accuracy (0.52%); however, the performance of Mutual Information based weight assignment was poor, only 12% more samples were removed and there was a 3% accuracy drop observed. Conversely, for the SUSY data set, the Mutual Information based weighted assignment provided a better performance, more than 15% samples were excluded while higher (0.98%) accuracy was also acquired. Although for the SUSY data set the PAC based weighted assignment approach surprisingly shrinks the data set to a very small size, the accuracy drop is larger than 6%. The results demonstrate the fact that, it is feasible to improve the performance of the proposed approach by assigning

weights for dimensions in the feature space. However, different approaches shoul d be considered for different data sets when determining the importance of each dimension.

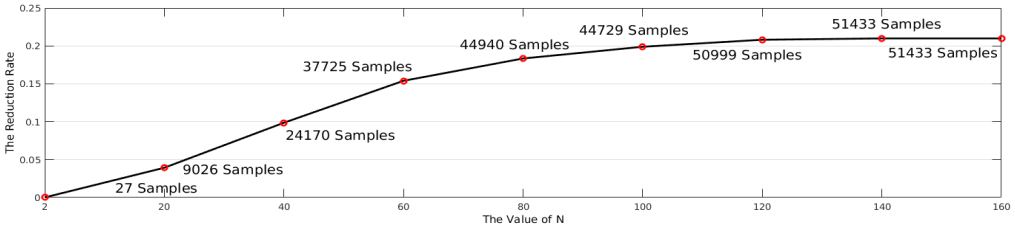
6.4 Conclusion

Finding a subset of large scale data set is useful for data mining scientists to process big data. However, it's difficult to measure the difference between true distribution and the subset distribution. This study aims to evaluate the difference between the original set and the subset according to the localized generalization error. By finding the error bounds, a Q-neighborhood based data reduction approach is proposed. The experimental results showed that for lower dimensional data sets, the proposed approach can effectively reduce the data set size, and there is little classification accuracy drop when employing the reduced data set to train the classifier.

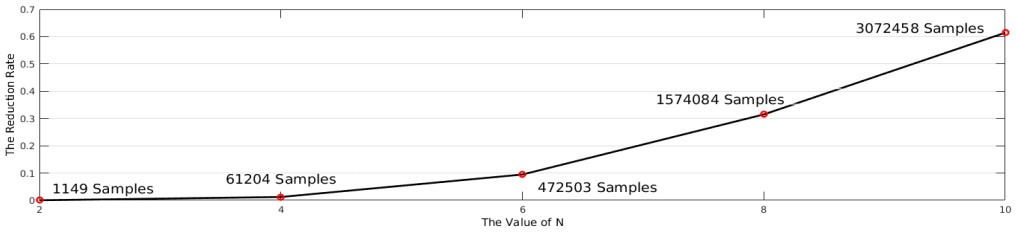
This approach is highly related the feature representation conditions of a certain application. For lower dimensional data sets with massive records, the proposed approach is a feasible choice for shrinking the size of the set efficiently and reliably. However, it is noted that the proposed method shall become inefficient for higher dimensional data sets (e.g. image data or bioinformatics data) and requires exponentially more computational resources to handle higher dimensional cases. One possible solution is to employ the proposed approach together with feature reduction algorithms. Our future study will mainly concentrate on tackling this limitation of the proposed approach.



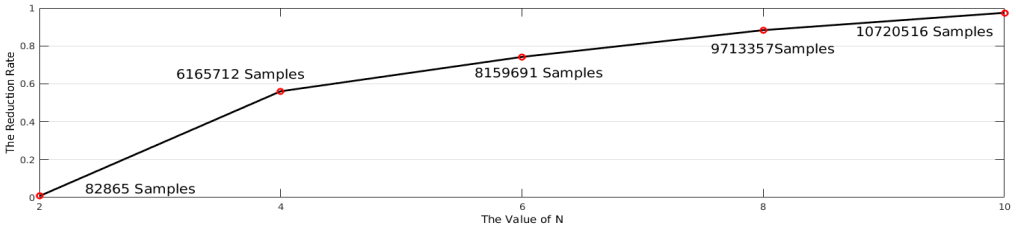
(a) Changing of reduction rate for Statlog



(b) Changing of reduction rate for Skin Segmentation

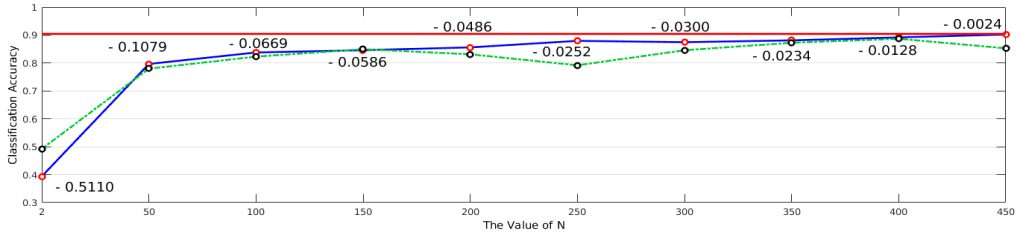


(c) Changing of reduction rate for SUSY

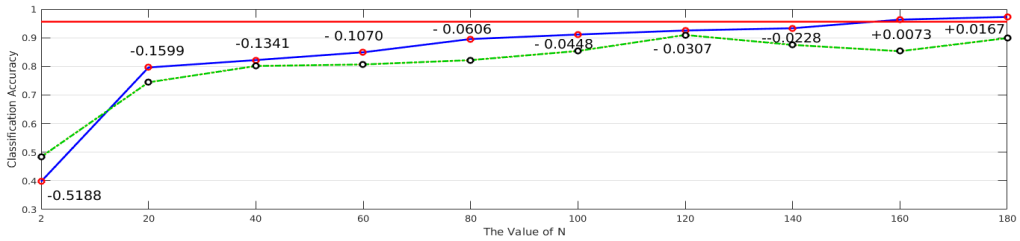


(d) Changing of reduction rate for HIGGS

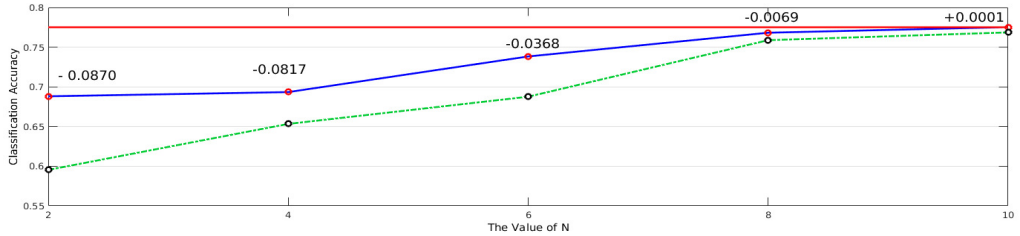
Figure 6.2. The reduction capacity of the proposed approach with different values of N



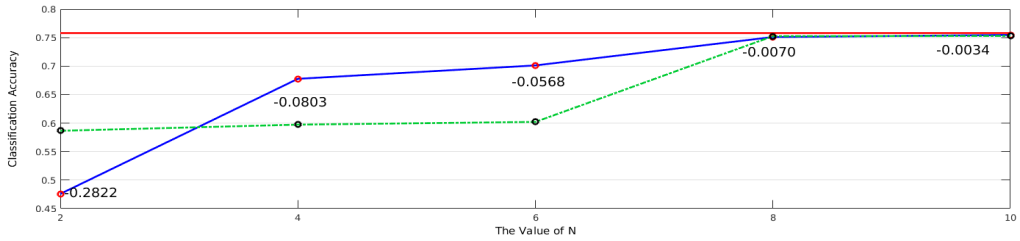
(a) Changing of classification accuracy for Statlog



(b) Changing of classification accuracy for Skin Segmentation



(c) Changing of classification accuracy for SUSY



(d) Changing of classification accuracy for HIGGS

Figure 6.3. The classification accuracies for each data set

Chapter 7

Conclusion

In this thesis, we have realized the following results.

- We developed algorithms for feature selection and feature learning for large scale data set, both for discrete and continuous values.
- We developed algorithms for improving the efficiency of feature learning in large scale data set.
- We presented algorithms for learning features from multiple-domain data sets.

Firstly we investigated feature selection for large scale data sets with discrete values. Our investigation focuses on the inconsistent condition (conflicting cases) of a certain data set which is called a DT in rough set theory. By analyzing the inconsistent condition of a certain DT in the VPRS model, we tried to find the reason why the reduct anomalies are generated. A new definition of an inconsistent condition in the VPRS model is proposed in order to investigate the reducts anomalies and consequently a new

definition of reduct in VPRS is given, which is a supplement of β-distribution reduct. This is a new approach to solve the feature selection problem in large scale data set.

Second, an efficient approach for feature learning by RBFNN was proposed. We introduced an approach to quickly determine the RBF centers for an RBFNN model. An eigenvector based clustering method was employed to calculate the RBF centers in the input feature space. RBF centers for the RBFNN model thus can be determined very quickly by calculating the principal components of the data matrix instead of the iterative calculation process of k-means clustering. After that, the connecting weights of the network can be easily obtained via either pseudo-inverse solution or the gradient descent algorithm. To evaluate the proposed approach, the performance of RBFNNs trained via different training schemes were compared in the experiments. It shows that the proposed method greatly reduces the training time of an RBFNN while allowing the RBFNN to attain a comparable accuracy result.

Third, we aimed to utilize multi-layer neural network on the feature representation of univariate and cross-domain time-series weather records. This study presentated an application about the feature representation issue of the given weather data sets from multiple meteorological domains including the temperature data, the atmospheric pressure data, and the wind speed data. In detail, firstly, several widely used multi-layer neural network architectures are tested on each kind of weather data set respectively, results in this stage of the study demonstrated the potential of multi-layer neural network for the feature representation on the time series univariate data sets. Consequently, to improve the forecasting accuracy of wind speed data in advance, the information from temperature data and atmospheric pressure data is utilized together with wind speed records to train up the forecasting models. Several models that can learn a fusing representation of time series data from different domains, including the proposed canonical correla-

tion analysis based Split-Autoencoder, are tested and compared. The employed computational intelligence models (e.g. Support Vector Machine) trained up in the learned fusing feature space are expected to provide higher prediction accuracy.

Fourth, a data reduction approach was proposed to acquire a representative subset of the large scale sample collections for learning tasks. Inspired by the localized generalization error theory, we provide a Q-neighborhood based data reduction approach. Specifically, based on a localized generalization error measure, the feature space is divided into Q-neighborhood grids. Representative samples are then selected from each Q-neighborhood grid. Since the localized generalization error bounds from the above the generalization error within the Q-neighborhood of the training samples, the obtained subset thus is expected to be able to approximate the real distribution with an acceptable error margin.

In this thesis, we mainly discussed the feature representation for different types of large-scale data. The multi-layer NNs now have become the most widely used approach for feature learning. Although multi-layer NNs, especially those with convolution structures are more suitable for the image and linguistic data, their revised versions may obtain success in for data sets from other domains.

This thesis explored the feature representation for data sets from other domains. both traditional approaches and deep learning approaches were all investigated. We have started some real-life machine learning tasks both use image data and numerical data in the medical field. Effectively representation of features in such a complex background will be my main research work in the future.

Chapter 8

Appendix

Set X', X'' are two equal-size data sets with different dimensions, $X' = (X'_1, \ldots, X'_N)^{\mathrm{T}}$ where $X'_i = (x'_1, \ldots, x'_{d_{X'}})$, $X'' = (X''_1, \ldots, X''_N)^{\mathrm{T}}$ where $X''_i = (x''_1, \ldots, x''_{d_{X''}})$, $i = 1, \ldots, N$. The main concept of CCA is to calculate the correlation between a linear combination of the variables in one data set and a linear combination of the variables in the other data set. Specifically, CCA seeks projection vectors a_1, b_1 such that $y'_1 = a_1^{\mathrm{T}} X'$ and $y''_1 = b_1^{\mathrm{T}} X''$ maximize the correlation $corr(y'_1, y''_1)$. y'_1 and y''_1 are called the first pair of canonical variables. Consequently, CCA seeks projection vectors a_2, b_2 and to maximize $corr(y'_2, y''_2)$ where $y'_2 = a_2^{\mathrm{T}} X'$, $y''_2 = b_2^{\mathrm{T}} X''$, and subject to the constraint that $a_1^{\mathrm{T}} X'$ and $a_2^{\mathrm{T}} X'$ are uncorrelated, $b_1^{\mathrm{T}} X''$ and $b_2^{\mathrm{T}} X''$ are uncorrelated. This gives the second pair of canonical variables. Such procedure may be continued up to $\min\{d_{X'}, d_{X''}\}$ times. The first k pairs of canonical variables are $Y' = (y'_1, \ldots, y'_k)$ and $Y'' = (y''_1, \ldots, Y''_k)$ and be solved via finding the eigenvectors of a certain matrix.

To seek a_1 and b_1, let $\Sigma_{X'X'} = \mathrm{cov}(X', X')$ and $\Sigma_{X''X''} = \mathrm{cov}(X'', X'')$, CCA needs

to maximize the correlation coefficient

$$\begin{aligned} \rho &= \frac{a_1^{\mathrm{T}}\Sigma_{X'X''}b_1}{\sqrt{a_1^{\mathrm{T}}\Sigma_{X'X'}a_1 b_1^{\mathrm{T}}\Sigma_{X''X''}b_1}}, \\ \text{s.t.} \quad & \mathrm{Var}(u_1) = a_1^{\mathrm{T}}\Sigma_{X'X'}a_1 = 1, \\ & \mathrm{Var}(v_1) = b_1^{\mathrm{T}}\Sigma_{X''X''}b_1 = 1. \end{aligned} \tag{8.1}$$

To solve Eq.8.1, two Lagrange multipliers λ' and λ'' can be introduced for constructing the Lagrange function

$$\begin{aligned} \mathcal{L}(a_1, b_1, \lambda', \lambda'') =& a_1^{\mathrm{T}}\Sigma_{X'X''}b_1 - \frac{\lambda'}{2}(a_1^{\mathrm{T}}\Sigma_{X'X'}a_1 - 1) \\ & -\frac{\lambda''}{2}(b_1^{\mathrm{T}}\Sigma_{X''X''}b_1 - 1). \end{aligned} \tag{8.2}$$

By giving the first partial derivatives of Eq.8.2, it can be optimized via

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial a_1} = \Sigma_{X'X''}b_1 - \lambda'\Sigma_{X'X'}a_1 = 0 \\ \dfrac{\partial \mathcal{L}}{\partial b_1} = \Sigma_{X''X'}a_1 - \lambda''\Sigma_{X''X''}b_1 = 0 \end{cases}. \tag{8.3}$$

After some algebra, there is

$$\Sigma_{X'X'}^{-1}\Sigma_{X'X''}\Sigma_{X''X''}^{-1}\Sigma_{X''X'}a_1 - \lambda'^2 a_1 = 0 \tag{8.4}$$

and

$$\Sigma_{X''X''}^{-1}\Sigma_{X''X'}\Sigma_{X'X'}^{-1}\Sigma_{X'X''}b_1 - \lambda'^2 b_1 = 0. \tag{8.5}$$

Therefore a_1 is the eigenvector of $\Sigma_{X'X'}^{-1}\Sigma_{X'X''}\Sigma_{X''X''}^{-1}\Sigma_{X''X'}$ and b_1 is the eigenvector of $\Sigma_{X''X''}^{-1}\Sigma_{X''X'}\Sigma_{X'X'}^{-1}\Sigma_{X'X''}$ w.r.t. the eigenvalue of λ'.

Consequently, a_2 and b_2 could be calculated via adding two more constraints for Eq.8.1 to make sure $\text{cov}(a_1^{\text{T}}X', a_2^{\text{T}}X') = a_1^{\text{T}}\Sigma_{X'X'}a_2 = 0$ and $\text{cov}(b_1^{\text{T}}X'', b_2^{\text{T}}X'') = b_1^{\text{T}}\Sigma_{X''X''}b_2 = 0$. In the way, $a_3, \vdots..., a_k$ can also be calculated.

In general, define $T = \Sigma_{X'X'}^{-\frac{1}{2}}\Sigma_{X'X''}\Sigma_{X''X''}^{-\frac{1}{2}}$, for a given k, the k pairs of projection vectors to maximize the correlation coefficients could be calculated via eigenvectors of TT^{T}. Set the top ith eigenvalue of TT^{T} is λ_i^2, $i = 1, \ldots, k$, u_i be the corresponding eigenvector, the ith pair of projection vectors a_i, b_i are

$$a_i = \Sigma_{X'X'}^{-\frac{1}{2}}u_i, \quad b_i = \lambda_i^{-1}\Sigma_{X''X''}^{-1}\Sigma_{X''X'}a_i, \tag{8.6}$$

and $y_i' = a_iX'$, $y_i'' = b_iX''$ is the ith canonical correlation variable while λ_i is the ith canonical correlation coefficients. The total canonical correlation of X' and X'' is the sum of the top k singular values of the matrix T, which is equal to $\text{tr}(TT^{\text{T}})^{\frac{1}{2}}$.

In the training processing of the CCA based split Autoencoder, to perform back-propagation to optimize Eq.5.19, the gradient of canonical correlation term should also be given. Let U_k and V_k be the left- and right- singular vectors of T, i.e. the singular value decomposition of T is $T = U_kDV_k^{\text{T}}$, the first partial derivative of total canonical correlation $corr(Y', Y'') = \text{tr}(TT^{\text{T}})^{\frac{1}{2}}$ w.r.t Y' is

$$\frac{\partial corr(Y', Y'')}{\partial Y'} = \frac{1}{N-1}(2\nabla_{11}\bar{Y}' + \nabla_{12}\bar{Y}''), \tag{8.7}$$

where

$$\nabla_{11} = -\frac{1}{2}\Sigma_{Y'Y'}^{-\frac{1}{2}}U_kDU_K^{\text{T}}\Sigma_{Y'Y'}^{-\frac{1}{2}}, \tag{8.8}$$

and

$$\nabla_{12} = \Sigma_{Y'Y'}^{-\frac{1}{2}}U_kV_K^{\text{T}}\Sigma_{Y''Y''}^{-\frac{1}{2}}, \tag{8.9}$$

The $\bar{Y}'$ and $\bar{Y}''$ are the centered matrices of Y' and Y'' respectively. The first partial derivative w.r.t. Y'' has a symmetric expression.

Bibliography

[1] D. ping Tian *et al.*, "A review on image feature extraction and representation techniques," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 8, no. 4, pp. 385–396, 2013.

[2] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[3] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature selection for svms," in *Advances in neural information processing systems*, pp. 668–674, 2001.

[4] P. Mitra, C. Murthy, and S. K. Pal, "Unsupervised feature selection using feature similarity," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 3, pp. 301–312, 2002.

[5] M. Ringnér, "What is principal component analysis?," *Nature biotechnology*, vol. 26, no. 3, pp. 303–304, 2008.

[6] J. B. Tenenbaum, "Mapping a manifold of perceptual observations," in *Advances in neural information processing systems*, pp. 682–688, 1998.

[7] V. N. Vapnik, *Statistical learning theory*, vol. 1. Wiley New York, 1998.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[9] Y. LeCun, L. Bottou, Y. Bengio, *et al.*, "Lenet-5, convolutional neural networks (2015)," *Retrieved June*, vol. 1, 2016.

[10] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern recognition letters*, vol. 15, no. 11, pp. 1119–1125, 1994.

[11] Y. Kim, W. N. Street, and F. Menczer, "Feature selection in unsupervised learning via evolutionary search," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 365–369, 2000.

[12] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with elms for big data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, 2013.

[13] B. Hssina, A. Merbouha, H. Ezzikouri, and M. Erritali, "A comparative study of decision tree id3 and c4. 5," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 2, pp. 13–19, 2014.

[14] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, "The cart decision tree for mining data streams," *Information Sciences*, vol. 266, pp. 1–15, 2014.

[15] C. H. Li and C. Lee, "Minimum cross entropy thresholding," *Pattern recognition*, vol. 26, no. 4, pp. 617–625, 1993.

[16] J. T. Kent, "Information gain and a general measure of correlation," *Biometrika*, vol. 70, no. 1, pp. 163–173, 1983.

[17] G. Deco and D. Obradovic, *An information-theoretic approach to neural computing*. Springer Science & Business Media, 2012.

[18] C. Rosenberg, M. Hebert, and S. Thrun, "Color constancy using kl-divergence," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, pp. 239–246, IEEE, 2001.

[19] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pp. 1200–1205, Ieee, 2015.

[20] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, "Supervised dictionary learning," in *Advances in neural information processing systems*, pp. 1033–1040, 2009.

[21] H. D. Beale, H. B. Demuth, and M. Hagan, "Neural network design," *Pws, Boston*, 1996.

[22] J. Ni, Q. Qiu, and R. Chellappa, "Subspace interpolation via dictionary learning for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 692–699, 2013.

[23] P. Comon, "Independent component analysis, a new concept?," *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[24] P. Zhao and B. Yu, "On model selection consistency of lasso," *Journal of Machine learning research*, vol. 7, no. Nov, pp. 2541–2563, 2006.

[25] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in Neural Information Processing Systems*, pp. 1096–1104, 2009.

[26] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[27] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv preprint arXiv:1803.01164*, 2018.

[28] J. N. Liu, Y. Hu, and Y. He, "A set covering based approach to find the reduct of variable precision rough set," *Information Sciences*, vol. 275, pp. 83–100, 2014.

[29] J. N. Liu, J. J. You, Y. Hu, and Y. He, "An advancing investigation on reduct and consistency for decision tables in variable precision rough set models," in *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1496–1503, IEEE, 2014.

[30] Y. Hu, J. J. You, J. N. Liu, and T. He, "An eigenvector based center selection for fast training scheme of rbfnn," *Information Sciences*, vol. 428, pp. 62–75, 2018.

[31] J. N. Liu, Y. Hu, Y. He, P. W. Chan, and L. Lai, "Deep neural network modeling for big data weather forecasting," in *Information Granularity, Big Data, and Computational Intelligence*, pp. 389–408, Springer, 2015.

[32] B. Kumari and T. Swarnkar, "Filter versus wrapper feature subset selection in large dimensionality micro array: A review," 2011.

[33] Z. Zhu, Y.-S. Ong, and M. Dash, "Wrapper–filter feature selection algorithm using a memetic framework," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 70–76, 2007.

[34] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data classification: Algorithms and applications*, p. 37, 2014.

[35] W. W. Ng, D. S. Yeung, M. Firth, E. C. Tsang, and X.-Z. Wang, "Feature selection using localized generalization error for supervised classification problems using rbfnn," *Pattern Recognition*, vol. 41, no. 12, pp. 3706–3719, 2008.

[36] A. Hajnayeb, A. Ghasemloonia, S. Khadem, and M. Moradi, "Application and comparison of an ann-based feature selection method and the genetic algorithm in gearbox fault diagnosis," *Expert systems with Applications*, vol. 38, no. 8, pp. 10205–10209, 2011.

[37] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[38] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[39] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[40] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[41] A. Krizhevsky, I. Sutskever, and G. Hinton, "'2012 alexnet," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Resnet-deep residual learning for image recognition," *ResNet: Deep Residual Learning for Image Recognition*, 2015.

[43] Z. Pawlak, J. Grzymala-Busse, R. Slowinski, and W. Ziarko, "Rough sets," *Communications of the ACM*, vol. 38, no. 11, pp. 88–95, 1995.

[44] W. Ziarko, "Variable precision rough set model," *Journal of computer and system sciences*, vol. 46, no. 1, pp. 39–59, 1993.

[45] A. An, N. Shan, C. Chan, N. Cercone, and W. Ziarko, "Discovering rules for water demand prediction: An enhanced rough-set approach," *Engineering Applications of Artificial Intelligence*, vol. 9, no. 6, pp. 645–653, 1996.

[46] W. Li, S. Chen, and B. Wang, "A variable precision rough set based modeling method for pulsed gtaw," *The International Journal of Advanced Manufacturing Technology*, vol. 36, no. 11, pp. 1072–1079, 2008.

[47] M. Inuiguchi, "Attribute reduction in variable precision rough set model," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 14, no. 04, pp. 461–479, 2006.

[48] K. Thangavel and A. Pethalakshmi, "Dimensionality reduction based on rough set theory: A review," *Applied Soft Computing*, vol. 9, no. 1, pp. 1–12, 2009.

[49] X. Zhang, Z. Mo, F. Xiong, and W. Cheng, "Comparative study of variable precision rough set model and graded rough set model," *International Journal of Approximate Reasoning*, vol. 53, no. 1, pp. 104–116, 2012.

[50] J. Wang and D. Miao, "Analysis on attribute reduction strategies of rough set," *Journal of computer science and technology*, vol. 13, no. 2, pp. 189–192, 1998.

[51] C. Rauszer and A. Skowron, "The discernibility matrices and functions in information systems," in *Intelligent Decision Support-Handbook of Applications and Advances of the Rough Sets Theory, Knowledge Engineering and Problem Solving*, vol. 11, (Norwell, MA, USA), pp. 331–362, Kluwer Academic Publishers, 1992.

[52] M. Beynon, "Reducts within the variable precision rough sets model: A further investigation," *European Journal of Operational Research*, vol. 134, no. 3, pp. 592 – 605, 2001.

[53] J. Mi, W. Wu, and W. Zhang, "Approaches to knowledge reduction based on variable precision rough set model," *Information Sciences*, vol. 159, no. 3, pp. 255–272, 2004.

[54] J. Wang and J. Zhou, "Research of reduct features in the variable precision rough set model," *Neurocomputing*, vol. 72, no. 10, pp. 2643–2648, 2009.

[55] J. Zhou and D. Miao, "β-interval attribute reduction in variable precision rough set model," *Soft Computing*, vol. 15, pp. 1643–1656, 2011.

[56] Z. Pawlak, "Rough sets and decision tables," in *Computation Theory* (A. Skowron, ed.), vol. 208 of *Lecture Notes in Computer Science*, pp. 187–196, Springer Berlin Heidelberg, 1985.

[57] Z. Pawlak, "Rough sets," *International Journal of Parallel Programming*, vol. 11, no. 5, pp. 341–356, 1982.

[58] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281–294, June 1989.

[59] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321–355, 1988.

[60] M. J. Powell, "Radial basis functions for multivariable interpolation: a review," in *Algorithms for Approximation*, pp. 143–167, Clarendon Press, 1987.

[61] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Networks*, vol. 14, no. 4, pp. 439–458, 2001.

[62] K. Mao, "RBF neural network center selection based on Fisher ratio class separability measure," *Neural Networks, IEEE Transactions on*, vol. 13, no. 5, pp. 1211–1217, 2002.

[63] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, 1990.

[64] C. M. Bishop, *Neural networks for pattern recognition*. Oxford University Press, 1995.

[65] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[66] T. Kohonen and P. Somervuo, "Self-organizing maps of symbol strings," *Neurocomputing*, vol. 21, no. 1, pp. 19–30, 1998.

[67] R. Saltos and R. Weber, "A rough–fuzzy approach for support vector clustering," *Information Sciences*, vol. 339, pp. 353–368, 2016.

[68] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.

[69] M. C. De Castro, C. De Castro, and D. S. Arantes, "Rbf neural networks with centers assignment via karhunen-loeve transform," in *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, vol. 2, pp. 1265–1270, IEEE, 1999.

[70] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *Neural Networks, IEEE Transactions on*, vol. 6, no. 6, pp. 1320–1329, 1995.

[71] C. P. Chen and J. Z. Wan, "A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 29, no. 1, pp. 62–72, 1999.

[72] S. Chen, C. F. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.

[73] F. Schwenker, H. A. Kestler, G. Palm, and M. Hoher, "Similarities of LVQ and RBF learning-a survey of learning rules and the application to the classification of signals from high-resolution electrocardiography," in *Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on*, vol. 1, pp. 646–651 vol.1, 1994.

[74] K. Jarrett, K. Kavukcuoglu, and Y. Lecun, "What is the best multi-stage architecture for object recognition?," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 2146–2153, IEEE, 2009.

[75] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, Oakland, CA, USA., 1967.

[76] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[77] C.-C. Liao, "Genetic k-means algorithm based rbf network for photovoltaic mpp prediction," *Energy*, vol. 35, no. 2, pp. 529–536, 2010.

[78] S.-K. Oh, W.-D. Kim, W. Pedrycz, and S.-C. Joo, "Design of k-means clustering-based polynomial radial basis function neural networks (pRBF-NNs) realized with the aid of particle swarm optimization and differential evolution," *Neurocomputing*, vol. 78, no. 1, pp. 121–132, 2012.

[79] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: applications and algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, 1999.

[80] M. H. Black and R. M. Watanabe, "A principal components-based clustering method to identify variants associated with complex traits," *Human Heredity*, vol. 71, no. 1, pp. 50–58, 2011.

[81] S. Meng, Y. Fu, T. Liu, and Y. Li, "Principal component analysis for clustering temporomandibular joint data," in *2015 8th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 1, pp. 422–425, IEEE, 2015.

[82] J. H. Na, M. S. Park, and J. Y. Choi, "Pre-clustered principal component analysis for fast training of new face databases," in *Control, Automation and Systems, 2007. ICCAS'07. International Conference on*, pp. 1144–1149, IEEE, 2007.

[83] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proceedings of the twenty-first International Conference on Machine Learning*, p. 29, ACM, 2004.

[84] H. Zha, X. He, C. Ding, M. Gu, and H. D. Simon, "Spectral relaxation for k-means clustering," in *Advances in neural information processing systems*, pp. 1057–1064, 2001.

[85] A. Gordon and J. Henderson, "An algorithm for euclidean sum of squares classification," *Biometrics*, pp. 355–362, 1977.

[86] K. Fan, "On a theorem of weyl concerning eigenvalues of linear transformations I," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 35, no. 11, p. 652, 1949.

[87] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1101–1113, 1993.

[88] D. Haussler, *Probably approximately correct learning*. University of California, Santa Cruz, Computer Research Laboratory, 1990.

[89] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 245–250, ACM, 2001.

[90] M. Alhamdoosh and D. Wang, "Fast decorrelated neural network ensembles with random weights," *Information Sciences*, vol. 264, pp. 104–117, 2014.

[91] A. Asuncion and D. J. Newman, "UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences." `http://archive.ics.uci.edu/ml/`. Accessed December 4, 2016.

[92] Y. Bengio, "Learning deep architectures for AI," *Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[93] T. Kohonen, "Improved versions of learning vector quantization," in *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pp. 545–550, IEEE, 1990.

[94] E. Kalnay and M. Cai, "Impact of urbanization and land-use change on climate," *Nature*, vol. 423, no. 6939, pp. 528–531, 2003.

[95] L. Kalkstein and K. Smoyer, "The impact of climate change on human health: some international implications," *Experientia*, vol. 49, no. 11, pp. 969–979, 1993.

[96] G. Gutman and A. Ignatov, "The derivation of the green vegetation fraction from noaa/avhrr data for use in numerical weather prediction models," *International Journal of Remote Sensing*, vol. 19, no. 8, pp. 1533–1543, 1998.

[97] S. Boussetta, G. Balsamo, A. Beljaars, T. Kral, and L. Jarlan, "Impact of a satellite-derived leaf area index monthly climatology in a global numerical weather prediction model," *International Journal of Remote Sensing*, vol. 34, no. 9-10, pp. 3520–3542, 2013.

[98] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Master-

ing the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[99] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[100] M. Lei, L. Shiyan, J. Chuanwen, L. Hongling, and Z. Yan, "A review on the forecasting of wind speed and generated power," *Renewable and Sustainable Energy Reviews*, vol. 13, no. 4, pp. 915–920, 2009.

[101] P. Li and E. S. Lai, "Short-range quantitative precipitation forecasting in Hong Kong," *Journal of Hydrology*, vol. 288, no. 1, pp. 189–209, 2004.

[102] J. D. Hamilton, *Time series analysis*, vol. 2. Princeton University Press Princeton, 1994.

[103] J. W. Taylor, L. M. De Menezes, and P. E. McSharry, "A comparison of univariate methods for forecasting electricity demand up to a day ahead," *International Journal of Forecasting*, vol. 22, no. 1, pp. 1–16, 2006.

[104] Ö. A. Dombaycı and M. Gölcü, "Daily means ambient temperature prediction using artificial neural network method: A case study of turkey," *Renewable Energy*, vol. 34, no. 4, pp. 1158–1161, 2009.

[105] S. P. Kani and M. Ardehali, "Very short-term wind speed prediction: a new artificial neural network–Markov chain model," *Energy Conversion and Management*, vol. 52, no. 1, pp. 738–745, 2011.

[106] H. Liu, H.-q. Tian, and Y.-f. Li, "Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction," *Applied Energy*, vol. 98, pp. 415–424, 2012.

[107] A. Sfetsos, "A comparison of various forecasting techniques applied to mean hourly wind speed time series," *Renewable Energy*, vol. 21, no. 1, pp. 23–35, 2000.

[108] Y. Bengio *et al.*, "Deep learning of representations for unsupervised and transfer learning.," *ICML Unsupervised and Transfer Learning*, vol. 27, pp. 17–36, 2012.

[109] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting*. Springer Science & Business Media, 2006.

[110] G. Li and J. Shi, "On comparing three artificial neural networks for wind speed forecasting," *Applied Energy*, vol. 87, no. 7, pp. 2313–2320, 2010.

[111] A. Ng, "Sparse autoencoder," *CS294A Lecture Notes*, vol. 72, pp. 1–19, 2011.

[112] H. Chen and A. F. Murray, "Continuous Restricted Boltzmann Machine with an implementable training algorithm," in *IEE Proceedings - Vision, Image and Signal Processing*, vol. 150, pp. 153–158, IET, 2003.

[113] B. J. Frey, "Continuous sigmoidal belief networks trained using slice sampling," in *Proceedings of the 9th Conference and Workshop on Neural Information Processing Systems (NIPS-1996)*, pp. 452–458, MIT Press, 1997.

[114] M. Mohandes, T. Halawani, S. Rehman, and A. A. Hussain, "Support Vector Machines for wind speed prediction," *Renewable Energy*, vol. 29, no. 6, pp. 939–947, 2004.

[115] K. Chen and J. Yu, "Short-term wind speed prediction using an unscented kalman filter based state-space Support Vector regression approach," *Applied Energy*, vol. 113, pp. 690–705, 2014.

[116] Y. Ren, P. N. Suganthan, and N. Srikanth, "A comparative study of empirical mode decomposition-based short-term wind speed forecasting methods," *IEEE Transactions on Sustainable Energy*, vol. 6, pp. 236–244, Jan 2015.

[117] C.-N. Lu, H.-T. Wu, and S. Vemuri, "Neural network based short term load forecasting," *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 336–342, 1993.

[118] J. Shukla and B. M. Misra, "Relationships between sea surface temperature and wind speed over the central arabian sea, and monsoon rainfall over India," *Monthly Weather Review*, vol. 105, no. 8, pp. 998–1002, 1977.

[119] G. D. Atkinson and C. R. Holliday, "Tropical cyclone minimum sea level pressure/maximum sustained wind relationship for the western north pacific," *Monthly Weather Review*, vol. 105, no. 4, pp. 421–427, 1977.

[120] Y. Zheng, "Methodologies for cross-domain data fusion: An overview," *IEEE Transactions on Big Data*, vol. 1, no. 1, pp. 16–34, 2015.

[121] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 689–696, 2011.

[122] W. Wang, R. Arora, K. Livescu, and J. Bilmes, "On deep multi-view representation learning," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1083–1092, 2015.

[123] K. V. Mardia, J. T. Kent, and J. M. Bibby, "Multivariate analysis," 1980.

[124] G. Andrew, R. Arora, J. A. Bilmes, and K. Livescu, "Deep canonical correlation analysis," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1247–1255, 2013.

[125] N. Tishby and N. Slonim, "Data clustering by markovian relaxation and the information bottleneck method," in *Proceedings of the 13rd Conference and Workshop on Neural Information Processing Systems (NIPS-2000)*, pp. 640–646, Citeseer, 2000.

[126] G. Chechik, A. Globerson, N. Tishby, and Y. Weiss, "Information bottleneck for gaussian variables," *Journal of Machine Learning Research*, vol. 6, no. Jan, pp. 165–188, 2005.

[127] W. Wang, R. Arora, K. Livescu, and J. A. Bilmes, "Unsupervised learning of acoustic features via deep canonical correlation analysis," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4590–4594, IEEE, 2015.

[128] T. G. Barbounis, J. B. Theocharis, M. C. Alexiadis, and P. S. Dokopoulos, "Long-term wind speed and power forecasting using local recurrent neural network models," *IEEE Transactions on Energy Conversion*, vol. 21, no. 1, pp. 273–284, 2006.

[129] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 26, no. 1, pp. 97–107, 2014.

[130] J. Hammersley, *Monte Carlo methods*. Springer Science & Business Media, 2013.

[131] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[132] A. Cuzzocrea, I.-Y. Song, and K. C. Davis, "Analytics over large-scale multidimensional data: the big data revolution!," in *Proceedings of the ACM 14th*

International Workshop on Data Warehousing and OLAP, pp. 101–104, ACM, 2011.

[133] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, "Efficient machine learning for big data: A review," *Big Data Research*, 2015.

[134] I. Czarnowski, "Cluster-based instance selection for machine classification," *Knowledge and Information Systems*, vol. 30, no. 1, pp. 113–133, 2012.

[135] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data mining and knowledge discovery*, vol. 6, no. 2, pp. 153–172, 2002.

[136] H. Liu and H. Motoda, "On issues of instance selection," *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 115–130, 2002.

[137] N. Jayaram and J. W. Baker, "Efficient sampling and data reduction techniques for probabilistic seismic lifeline risk assessment," *Earthquake Engineering & Structural Dynamics*, vol. 39, no. 10, pp. 1109–1131, 2010.

[138] D. R. Wilson and T. R. Martinez, "An integrated instance-based learning algorithm," *Computational Intelligence*, vol. 16, no. 1, pp. 1–28, 2000.

[139] F. Angiulli, "Fast condensed nearest neighbor rule," in *Proceedings of the 22nd international conference on Machine learning*, pp. 25–32, ACM, 2005.

[140] G. Ritter, H. Woodruff, S. Lowry, and T. Isenhour, "An algorithm for a selective nearest neighbor decision rule," *IEEE Transactions on Information Theory*, vol. 21, no. 6, pp. 665–669, 1975.

[141] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine learning*, vol. 38, no. 3, pp. 257–286, 2000.

[142] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine learning*, vol. 6, no. 1, pp. 37–66, 1991.

[143] S. Eschrich, J. Ke, L. O. Hall, and D. B. Goldgof, "Fast accurate fuzzy clustering through data reduction," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 2, pp. 262–270, 2003.

[144] L. I. Kuncheva and J. C. Bezdek, "Nearest prototype classification: clustering, genetic algorithms, or random search?," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 28, no. 1, pp. 160–164, 1998.

[145] Y. Hamo and S. Markovitch, "The compset algorithm for subset selection.," in *IJCAI*, pp. 728–733, 2005.

[146] H. Zhang and G. Sun, "Optimal reference subset selection for nearest neighbor classification by tabu search," *Pattern Recognition*, vol. 35, no. 7, pp. 1481–1490, 2002.

[147] J.-H. Chen, H.-M. Chen, and S.-Y. Ho, "Design of nearest neighbor classifiers: multi-objective approach," *International Journal of Approximate Reasoning*, vol. 40, no. 1, pp. 3–22, 2005.

[148] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 561–575, 2003.

[149] S. García, J. R. Cano, and F. Herrera, "A memetic algorithm for evolutionary prototype selection: A scaling up approach," *Pattern Recognition*, vol. 41, no. 8, pp. 2693–2709, 2008.

[150] D. S. Yeung, W. W. Ng, D. Wang, E. C. Tsang, and X.-Z. Wang, "Localized generalization error model and its application to architecture selection for radial basis function neural network," *Neural Networks, IEEE Transactions on*, vol. 18, no. 5, pp. 1294–1305, 2007.

[151] V. N. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[152] D. Haussler, "Overview of the probably approximately correct (pac) learning framework," *Information and Computation*, vol. 100, no. 1, pp. 78–150, 1992.

[153] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1963.

[154] P. P. Chan, D. S. Yeung, W. W. Ng, C. M. Lin, and J. N. Liu, "Dynamic fusion method using localized generalization error model," *Information Sciences*, vol. 217, pp. 1–20, 2012.

[155] N. Kwak and C.-H. Choi, "Input feature selection by mutual information based on parzen window," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 12, pp. 1667–1671, 2002.