UNIVERSITY OF TECHNOLOGY SYDNEY

Transfer Learning in Takagi-Sugeno Fuzzy Models

Hua Zuo

Faculty of Engineering and Information Technology

University of Technology Sydney

A thesis submitted for the Degree of

Doctor of Philosophy

May 2018

Certificate of Authorship/Originality

Acknowledgements

I would like to express my sincere gratitude to my principal supervisor, Professor Guangquan Zhang, and my co-supervisors, Professor Jie Lu and Dr Vahid Behbood. Their comprehensive guidance covered all aspects of my PhD study, including research methodology, research topic selection, experiments, academic writing skills and thesis writing, even sentence structure and formulas. Their critical comments and suggestions strengthened my study significantly. Their strict academic attitude and respectful personalities benefited my PhD study and will be a great treasure throughout my life. Without their excellent supervision and continuous encouragement, this research would not have been finished on time. Thanks to all of you for your kind help!

I am grateful to all members of the Decision Systems and e-Service Intelligent (DeSI) Lab in the Centre for Artificial Intelligence (CAI) for their careful participation in my presentation and their valuable comments on my research. I wish to express my appreciation to all friends (especially Hongshu Chen, Qian Zhang, Shan Xue and Fan Dong) for their help, accompany and valued comments in not only on my study, but also on my life. I would like to thank Ms Sue Felix and Ms Jem Moore for helping me to proofread my publications and this thesis.

I wish to express my appreciation for the financial support I received for my study. Special thanks go to the University of Technology Sydney (UTS) and the China Scholarship Council (CSC). Also thanks for the support of the Australian Research Council under DP 140101366.

Last but not least, I would also like to thank my family members. Thanks to my mother, father and my brother for their continued encouragement and generous support. Thank you for supporting me and always being there for me during my ups and downs. Pursuing a PhD has always been a long-term challenge for me and a dream of my brother’s. This dream would have not been achieved without the love, great empathy and kind assistance of my family.

Abstract

In classical data-driven machine learning methods, massive amounts of labeled data are required to build a high-performance prediction model. However, the amount of labeled data in many real-world applications is insufficient, so establishing a prediction model is impossible. Transfer learning has recently emerged as a way of exploiting previously acquired knowledge to solve new yet similar problems much more efficiently and effectively. It exploits the knowledge accumulated in auxiliary domains to help construct prediction models in a target domain with inadequate training data. Most existing methods on transfer learning address classification tasks, but studies on transfer learning in the case of regression problems, which is an important category in prediction models, are still scarce. In addition, the existing works ignore the inherent phenomenon of uncertainty - a crucial factor during the knowledge transfer process.

To fill these gaps, this research develops algorithms and methods to deal with transfer learning in homogeneous and heterogeneous spaces using fuzzy rule-based models. Fuzzy rules are first generated from the source domain through a learning process. These rules, as acquired knowledge, are then transferred to the target domain. First, a novel fuzzy rule-based domain adaptation method is proposed to transfer knowledge between domains in homogeneous spaces. It utilizes the existing fuzzy rules of source domain and modifies the input space with nonlinear mappings to adapt to the current regression tasks in the target domain. Second, a granular fuzzy domain adaptation framework, comprising three methods, is developed to handle the knowledge transfer problems based on the idea of granular computing. Thirdly, a fuzzy domain adaptation method is developed to handle the case that the numbers of fuzzy rules in two domains do not match. In addition, the proposed methods are also used to solve the classification problems in transfer learning. Fourthly, an innovative method that combines an infinite Gaussian mixture model with active learning is presented to discover the structure of data and actively augment information in a target domain. Fifthly, a fuzzy heterogeneous domain adaptation method is proposed to transfer knowledge in heterogeneous spaces. The proposed algorithms and methods are validated in each step of development using experiments performed on both synthetic and real-world datasets. The results show that this study significantly improve the performance of existing models when solving new tasks in the target domain in both homogeneous and heterogeneous domain adaptation settings.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

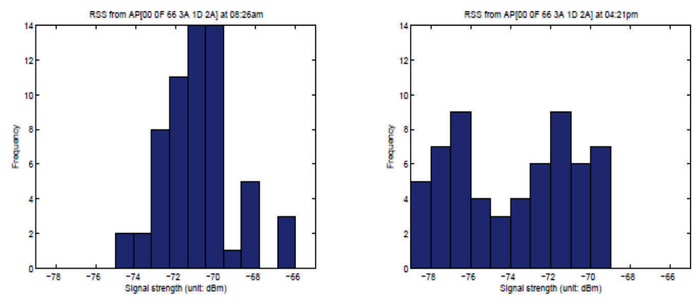
Traditional machine learning methods use learning models to extract knowledge from massive amounts of labeled data. They work under a common assumption that the training data (in the source domain) and the testing data (in the target domain) have the same feature space and the same probability distributions. However, if the feature space or the distribution of the target data changes, the models built from the source data become unsuitable and a new model needs to be rebuilt and trained from scratch. Additionally, if there is insufficient labeled target data, a new prediction model for the target data will be impossible to establish. This severely impedes the capacity of these methods to model in such environments.

What makes the human learning process advanced is our ability to transfer knowledge from one situation to another. Humans often draw upon more than just training for generalization: we continuously adapt to changing environments. Instead of learning from scratch, humans tackle new tasks on the basis of previously accumulated knowledge, and it is this unique ability that has inspired the application of transfer learning as a possible solution for the issue described above.

In recent years, research has increased the focus on transfer learning (Pan and Yang, 2010) and its application to real-world problems in the field of computational intelligence. As a possible solution to a lack of enough data with labels, transfer learning aims to construct new predictive models much more quickly and effectively by exploiting the knowledge accumulated in an auxiliary domain, which in some way, or to some extent, relates to the original domain. Transfer learning examples include: using already-categorized French documents to help classify English documents, building recognition models capable of identifying novel visual categories (Zhu et al., 2011), and predicting the banks' status in one country based on the data of banks in another country (Behbood et al., 2015).

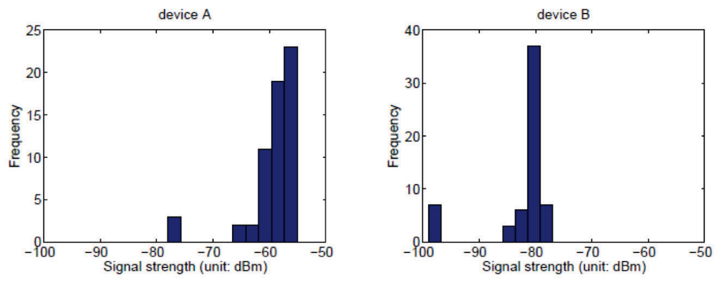
A well-known example of transfer learning is indoor location estimation through the WiFi signals (Lim et al., 2007). Location estimation in an indoor environment is an important task in AI, from activity recognition and robotics to various user-assisted technologies such as home-based healthcare. The WiFi-based indoor localization problem, however, is a difficult task because the data distribution is constantly changing depending on various factors, such as human movement, temperature and humidity. When applying machine-learning-based approaches, it is very costly to collect and label the training data in the form of received signal strength values (RSS) and location label pairs in a large-scale building, as this necessitates the use of a mobile device while walking through a building to collect the RSS values and mark down the ground truth locations. When the signal distribution changes, the processes has to be repeated again. For example, in Fig. 1.1, all the RSS values in all locations of a building are collected at 8:26 am, thus a well-performing location prediction model can be built. However, the RSS values at 4:21 pm are quite different from those at 8:26

am, so the previously constructed model cannot be used to predict the location using the RSS data obtained at 4:21 pm. Similarly, in Fig. 1.2, all the RSS values are collected by device A and a location estimation model can be built based on these data. If the device is changed, the model cannot be used anymore, i.e. a model built using data from device A cannot be used to estimate the location based on the data collected by device B. Transfer learning can handle the problem of detecting a user's current location based on previously collected WiFi data (Pan et al., 2008). To handle the adaptation problems in Fig. 1.1 and Fig. 1.2, transfer learning methods explore the knowledge learned in the previous time period or by the previous device, and abstract and transfer the obtained knowledge to the current domain (time period and new device) and build a new model to solve the new tasks.



(a) RSS at 08:26am (b) RSS at 04:21pm

Figure 1.1 RSS collected in different time periods



(a) RSS at device A (b) RSS at device B

Figure 1.2 RSS collected by different devices

Transfer learning sits within the machine learning research area; hence, its methods use many notable machine learning techniques as basic training models, such as **s**upport **v**ector **m**achine (SVM) (Xu et al., 2014), NN (Long et al., 2016), naïve Bayes (Gönen and Margolin, 2014), and case-based models (Klenk and Forbus, 2009). Additionally, researchers in deep learning explore the transfer ability of deep models (Bengio, 2012). In practice, it is common to pre-train a ConvNet on very large datasets, and remove the last fully-connected layers, then treat the rest of the ConvNet as a fixed feature extractor for the new datasets. This is motivated by the observation that the earlier features of a ConvNet contain more generic features (e.g. edge detector or color blob detector) that should be useful to many tasks, but later layers of the ConvNet become progressively more specific to the details of the classes contained in the original dataset. For more information on transfer learning, we refer the readers to several survey papers that provide reviews and summaries of the various transfer learning methods and categories. There have been many studies in the area of transfer learning, and related work can be divided into two categories: homogeneous domain adaptation and heterogeneous domain adaptation. In homogeneous domain adaptation, the source domain and target domain have the same feature space but different distributions. In heterogeneous domain adaptation, both the feature space and distributions are different in the source and target domains.

Although transfer learning exhibits an upward trend, there is still a huge gap between the existing work and domain adaptation tasks. For instance, a significant amount of transfer learning research has been undertaken for classification problems (Dai et al., 2007c), yet studies on regression problems are still scarce. In addition, the existing works ignore the inherent phenomenon of uncertainty – a crucial factor during the knowledge transfer process. There is a clear co-dependency between the level of certainty in learning a task and the amount of information that is available; problems with too little information have a high degree of uncertainty. If there are too few data with labels in the target domain, only a finite amount of information can be extracted, and this leads to a high degree of uncertainty. However, the introduction of fuzzy systems (Berkan and Trubatch, 1997) has shown promising results in overcoming this problem. Another reason why the current transfer learning methods can not be acceptaned is the information granularity inherent in many problems. For instance, 128GB of mobile storage is considered large today, whereas 32GB was regarded as large five years ago. The precise values, 128GB and 32GB, both need to be expressed as a granular value, “large”, for learning to be effectively transferred from the five-year-old domain to today’s domain. Extracting additional abstract knowledge shared between domains should therefore assist knowledge transfer. Granular computing (GrC) (Bargiela and Pedrycz, 2016) is an emerging information processing paradigm that transforms complex data into information granules at different levels of resolution to reveal different features and irregularities. GrC’s ability to address information at different levels of abstraction could improve the performance of transfer learning. This thesis intends to fill this gap by developing a set of fuzzy rule-based transfer learning methods for the domain adaptation problems.

1.2 Research questions and objectives

This research aims to develop a set of fuzzy transfer learning models to handle the domain adaptation problems in homogeneous and heterogeneous spaces, and will answer the following research questions:

QUESTION 1 *How to explore the relation between the source domain and target domain?*

The model of the source domain cannot be used to solve the tasks in the target domain mainly due to the discrepancy in feature space and the data distributions. A different feature space represents a different number of input variables, and non-identical distributions of two domains indicate the bias or poor performance when predicting the output of target data using the model of the source domain. However, the source domain is assumed to be related to the target domain, and the knowledge of the source domain could help improve the construction of the target model. Therefore, it is crucial to find out the the relation between the source and target domains and apply the relation to guide the knowledge transfer between two domains.

QUESTION 2 *How to grasp abstract knowledge in two domains?*

The abstract knowledge in two domains comprises two aspects: information granularity and abstract features. The granules in different platforms may have different representations, and the level of granularity is dependent on the given data. In transfer learning problems, it is important to find the appropriate form of granules and determine the granularity level of data, so that the abstract knowledge shared between domains can be transferred to a suitable level of granularity. Additionally, for the heterogeneous domain adaptation, the different feature spaces require exploring a more abstract space to represent the shared features, so the common knowledge in two domains can be delivered in an abstract way.

QUESTION 3 *How to utilize the existing fuzzy rules of the source domain to adapt to the tasks of the target domain in homogeneous space?*

A fuzzy rule consists of two components: a condition part, which is determined by the centers of the clustering, and a conclusion part, which is governed by the linear functions. The divergence of data's distributions leads to the different fuzzy rules conditions and non-identical domain knowledge influences the conclusions of fuzzy rules. However, the homogeneous space decides whether the fuzzy rules in two domains have the same dimension, and their feature spaces are the same. In order to apply the fuzzy rules of the souce domain for the target data, the fuzzy rules need to be changed, and the way the existing fuzzy rules are changed greatly affects the effectiveness of transfer learning.

QUESTION 4 *How to transfer the knowledge of the source domain to the target domain in heterogeneous space?*

The greatest divergence between the source domain and target domain is the difference in feature spaces. The non-identical dimensions mean that the model of the source domain cannot be used for target data. Thus, exploring a latent feature space is an important procedure to unify the features in two domains. With the transformation to the latent feature space, the domain adaptation problem converts from the homogeneous space to the heterogeneous space. In the latent feature space, although the number of features is the same for the two domains, the meanings of the feature and the distributions are still quite different. Thus, the fuzzy rules of the source domain still cannot be directly used for the target data. In addition, the relation between the fuzzy rules in the two domains become unclear after the data are projected to the latent feature space. Therefore, modifying of the fuzzy rules is still a challenging procedure to transfer the knowledge.

This research aims to achieve the following objectives, which are expected to answer the above research questions:

OBJECTIVE 1 *To discover the discrepancy between the fuzzy rule models in two domains.*

This objective corresponds to research Question 1. Since the fuzzy rule-based model is used in this thesis as the basic model for transfer learning, the divergence of the source and target domains is discussed from the perspective of fuzzy rules. The discrepancies between the fuzzy rules in the two domains are explored from two aspects. First, the feature spaces of the two domains are compared and the distributions are measured to determine whether the condition parts of the fuzzy rules is the same in the two domains. Secondly, the consistency of the fuzzy rules' conclusions is judged from the domain-related knowledge. Similar domain knowledge represents similar conditions of the fuzzy rules. This study develops a method to discover the relation between the source and target domains, and provides the basis for the following knowledge transfer procedure.

OBJECTIVE 2 *To develop a granular domain adaptation method to extract the abstract knowledge in the two domains.*

This objective corresponds to research Question 2. The fuzzy rules are regarded as information granules to represent the abstract knowledge and be transferred between the domains with different levels of granularity. A latent feature space is extracted to map the original data to the new feature space and convert the heterogeneous domain adaptation problem to the homogeneous problem. The learned latent feature space should satisfy several requirements: reducing the gap between the distributions of data, and keeping the manifold of data in the new feature space. This study develops a feature extraction method to extract a common feature space and proposes a granular domain adaptation method to facilitate the knowledge transfer through fuzzy rules.

OBJECTIVE 3 *To develop a fuzzy domain adaptation method to transfer knowledge between domains in homogeneous spaces.*

This objective corresponds to research Question 3. For a well-built fuzzy rules model, the useful components are the clusters' centers and the corresponding linear functions. In order to fit the target data, the existing centers of the clusters and coefficients of the linear functions need to be modified, and the labeled target data can be used to guide this procedure. This study develops an optimization-based method to modify the existing fuzzy rules of the source domain to fit the tasks in the target domain.

OBJECTIVE 4 *To develop a fuzzy domain adaptation method to transfer knowledge between domains in heterogeneous spaces.*

This objective corresponds to research Question 4. A latent feature space is learned to explored the abstract features shared between domains. The source and target data are mapped to a common latent feature space and share the same features. Although the relation between the fuzzy rules of the two domains in the latent feature space is not clear, we adopt the strategy of changing both the conditions and conclusions of the fuzzy rules, and apply the optimization method to guide the modification of the fuzzy rules. This study develops a fuzzy rule-based method to bridge the fuzzy rules of the two domains in heterogeneous space.

1.3 Research contributions

The main contributions of this study are concisely summarised as follows:

- A new fuzzy rule-based domain adaptation method is developed to transfer the knowledge of the source domain in homogeneous space by modifying the input space using nonlinear mappings, so that the existing fuzzy rules are more compatible with the target data.

- Granular domain adaptation methods are proposed to improve the performance of domain adaptation by considering the information granularity inherent in transfer learning. Further, an entire framework is presented to provide guidance for domain adaptation in the Takagi-Sugeno fuzzy models.

- A new domain adaptation method is proposed to transfer fuzzy rules from the source domain to the target domain even if the numbers of fuzzy rules does not match in two domains, which is a common case in the transfer learning problems.

- The knowledge contained in the unlabeled target data is explored to assist in the construction of the target model, which introduces a new way to extract as much as information as possible in the target domain.

- An innovative method that combines IGMM and active learning is presented to enhance the performance and generalizability of the construct model. IGMM is used to explore the relationship between the data structures in the source and target domains, and the idea of active learning is applied to increase the amount of labeled information in target domain by actively labeling the most informative data.

- A fuzzy domain adaptation method is developed to minimize the gap between the source domain and target domain, and facilitate the transfer of fuzzy rules for the two domains in heterogeneous spaces, which addresses the challenge brought about by the different feature spaces and distributions.

1.4 Research significance

The theoretical and practical significance of this research is summarized as follows:

Theoretical significance: The findings of this study contribute to the transfer learning community in the following two ways: enriching the theoretical analysis for transfer learning; and facilitating the application of transfer learning to more complex scenarios. More specifically, fuzzy systems are introduced to domain adaptation to handle the uncertainty during the transfer learning procedure, and the proposed methods could develop the capability of fuzzy rule-based methods in dealing with knowledge transfer between domains. Furthermore, the proposed methods concentrate on solving the regression tasks in transfer learning. Compared with the classification prediction problems, the continuous outputs in regression tasks are affected by more factors and are more complex. The extended application scenarios not only improve the impact of this area, but also motivate its continuing development.

Practical significance: The findings of this study contribute to the benefit of society given the important role transfer learning plays in modern life. The findings help resolve the real-world cold start problem especially in new areas where labeled data are scarce while a large amount of unlabeled data in related domains are available. Corporations and organizations can utilize the previous data or data from the related areas to help solve the current tasks although the data distributions and types are quite different, which saves a lot of human and financial resources. This has potential for many other applications that could benefit from this study.

1.5 Thesis structure

The structure of the thesis is shown in Fig. 1.3 and the chapters are organized as follows:

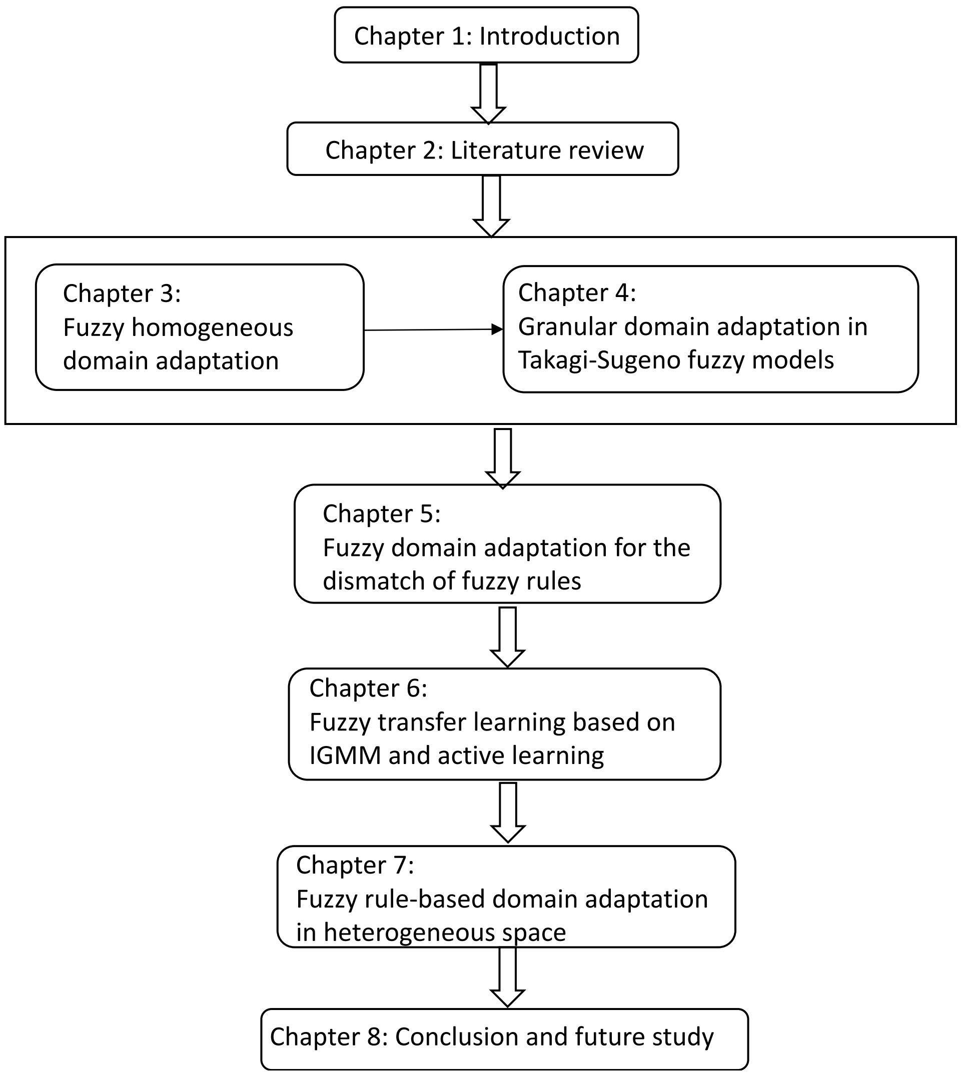

Chapter 1: Introduction
Chapter 2: Literature review
Chapter 3: Fuzzy homogeneous domain adaptation
Chapter 4: Granular domain adaptation in Takagi-Sugeno fuzzy models
Chapter 5: Fuzzy domain adaptation for the dismatch of fuzzy rules
Chapter 6: Fuzzy transfer learning based on IGMM and active learning
Chapter 7: Fuzzy rule-based domain adaptation in heterogeneous space
Chapter 8: Conclusion and future study

Figure 1.3 Thesis structure

- In Chapter 2, this thesis reviews the definitions related to transfer learning and the current state-of-the-art of the research on transfer learning. In addition, the idea of granular computing is introduced.

- In Chapter 3, this thesis develops a fuzzy rule-based transfer learning method to deal with the domain adaptation problems in homogeneous spaces. The input space is modified through mappings so that the existing fuzzy rules of the source domain are compatible with the current prediction tasks in the target domain.

- In Chapter 4, this thesis develops a granular transfer learning method to handle the domain adaptation problems in homogeneous space. Granular computing techniques are applied to help knowledge transfer between domains. Three methods and related algorithms are proposed, which considered together, constitute an overall framework that provides a comprehensive framework for the domain adaptation-based fuzzy models.

- In Chapter 5, this thesis presents a fuzzy domain adaptation method to solve the mismatching problems of the number of fuzzy rules in the two domains. In addition to the nonlinear mappings to change the input space, a piece-wise linear function is also used to construct the mappings and modify the existing fuzzy rules. In addition, the proposed method is used to solve domain adaptation in classification problems.

- In Chapter 6, this thesis presents a innovative methods that applies IGMM and active learning to enhance the performance and generalizability of the constructed model. IGMM is used to explore the relationship between the data structures in the source and target domains and provide guidance on a domain selection and transfer strategy. The idea of active learning is applied to increase the amount of labeled information in target domain by actively labeling the most informative data in the source domain for use in the target domain.

- In Chapter 7, this thesis presents a fuzzy regression transfer learning method to deal with the domain adaptation problems in heterogeneous spaces, where both the distributions and feature space are different. A latent feature space is extracted to minimize the gap between the two domains so that the knowledge can be transferred in the latent feature space.
- Finally, Chapter 8 summarises the findings of the thesis and points to directions for future work.

1.6 Publications related to this thesis

Following is a list of the refereed international journal and conference papers produced during my PhD research that have been published or are currently under review:

Publications in journals :

1. **Hua Zuo**, Guangquan Zhang, Witold Pedrycz, Vahid Behbood, and Jie Lu, 'Fuzzy regression transfer learning in Takagi-Sugeno fuzzy models', *IEEE Transactions on Fuzzy Systems*, Vol. 25, No. 6, pp. 1795-1807, 2016. (ERA rank: A*)
2. **Hua Zuo**, Guangquan Zhang, Witold Pedrycz, Vahid Behbood, and Jie Lu, 'Granular fuzzy regression domain adaptation in Takagi-Sugeno fuzzy models', *IEEE Transactions on Fuzzy Systems*, 2017, accepted. (ERA rank: A*)
3. Jie Lu, Vahid Behbood, Peng Hao, **Hua Zuo**, Shan Xue, and Guangquan Zhang, 'Transfer learning using computational intelligence: a survey', *Knowledge-Based Systems*, vol. 80, pp. 14-23, 2015. (ERA Rank: B)

4. **Hua Zuo**, Jie Lu, Guangquan Zhang, and Witold Pedrycz, 'Fuzzy rule-based domain adaptation in homogeneous and heterogeneous spaces', *IEEE Transactions on Fuzzy Systems*, under review.

5. **Hua Zuo**, Jie Lu, Guangquan Zhang, and Feng Liu, 'Fuzzy transfer learning based on infinite Gaussian mixture model and active learning', *IEEE Transactions on Fuzzy Systems*, submitted.

Publications in conferences :

6. **Hua Zuo**, Guangquan Zhang, Witold Pedrycz, Vahid Behbood, and Jie Lu, 'Fuzzy rule-based transfer learning for label space adaptation', *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2017. (ERA rank: A)

7. **Hua Zuo**, Guangquan Zhang, Witold Pedrycz, Vahid Behbood, and Jie Lu, 'Fuzzy Transfer Learning in Data-Shortage and Rapidly Changing Environments', *Proceedings of the 12th International FLINS Conference (FLINS)*, 2016. (ERA rank: B)

8. **Hua Zuo**, Guangquan Zhang, Witold Pedrycz, Vahid Behbood, and Jie Lu, 'Transfer learning in hierarchical feature spaces', *10th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pp. 183-188, 2015. (ERA rank: B)

9. **Hua Zuo**, Guangquan Zhang, Witold Pedrycz, Vahid Behbood, and Jie Lu, 'Feature Spaces-based Transfer Learning', *16th World Congress of the International Fuzzy Systems Association (IFSA)*, 2015. (ERA rank: B)

10. **Hua Zuo**, Guangquan Zhang, and Jie Lu, 'Semi-supervised domain adaptation in Takagi-Sugeno fuzzy rule models', *Proceedings of*

the 13th International FLINS Conference (FLINS), 2018, submitted. (ERA rank: B)

Chapter 2

Literature Review

This chapter presents a discussion of the research background and relevant work in connection with this research. In Section 2.1, the related definitions of transfer learning and domain adaptation are introduced. Sections 2.2 to 2.3 review the studies on transfer learning based on different ways of classifying the literature. Sections 2.2 presents the transfer learning techniques based on the content transferred between domains, including instance, feature representation, parameters, and relational knowledge. Section 2.3 discusses the transfer learning methods based on the domain knowledge, including supervised transfer, semi-supervised transfer, and unsupervised transfer. Section 2.4 reviews the literature on transfer learning based on the applied computational intelligence techniques, including NN, Bayes, fuzzy systems, and genetic algorithms. Section 2.5 introduces some basic ideas in granular computing. Section 2.6 briefly summarizes this chapter.

2.1 Definitions and categories of transfer learning and the category

The definitions related to transfer learning are introduced.

Definition 2.1. (Domain) (Pan and Yang, 2010)

A domain is denoted by $D = \{F, P(X)\}$, where F is a feature space, and $P(X), X = x_1, x_2, ..., x_n$ are the probability distributions of the instances.

Definition 2.2. (Task) (Pan and Yang, 2010)

A task is denoted by $T = \{Y, f(\cdot)\}$, where $Y \in R$ is the response, and $f(\cdot)$ is an objective predictive function.

Definition 2.3. (Transfer Learning) (Pan and Yang, 2010)

Given a source domain D_s, a learning task T_s, a target domain D_t, and a learning task T_t, transfer learning aims to improve the learning of the target predictive function $f_t(\cdot)$ in D_t using the knowledge in D_s an T_s, where $D_s \neq D_t$, or $T_s \neq T_t$.

Transfer learning addresses the problem of how to leverage previously acquired knowledge (a source domain) to improve the efficiency of learning in a new domain (the target domain). The existing methods in transfer learning can be mainly divided into two categories: homogeneous domain and adaptation and heterogeneous domain adaptation.

Definition 2.4. (Homogeneous Domain Adaptation)

Homogeneous domain adaptation is a category of transfer learning in which the feature space is the same, $F_s = F_t$, but the corresponding probability distributions are different $P_s(\boldsymbol{X}) \neq P_t((X))$.

Definition 2.5. (Heterogeneous Domain Adaptation)

Heterogeneous domain adaptation is a category of transfer learning in which the feature space is the same, $F_s \neq F_t$, but the corresponding probability distributions are different $P_s(\boldsymbol{X}) \neq P_t((X))$.

2.2 Different types of transfer learning methods based on content

The methods of transfer learning can be divided into four types: instance transfer learning, feature representation transfer learning, parameter transfer learning, and relational knowledge transfer learning. The main idea and the representative method in each category are described as follows.

The idea of instance transfer learning involves reweighting the instances of the source domain to ensure the new data have a similar distribution to the target domain, so that the model of the source domain can be used to solve the tasks in the target domain. The commonly used and well-known instance transfer learning method is a boosting algorithm, TrAdaBoost (Zhu, 2006). In TrAdaBoost, it is assumed that the data in the source and target domains have identical features and labels, but different marginal probability distributions. Due to the different distributions, some of the target data are helpful for learning the target model, but some are useless or even harmful to the learning. A method is proposed to change the weights of the data in the two domains with different strategies. The weights of the source data that have a positive effect are increased and the weights of the source data that have a negetive effect are decreased. Except for the data in the target domain, the weights associated with the incorrectly classified examples are increased, and the weights associated with the correctly classified examples are decreased.

The intuitive idea behind feature representation transfer learning is to learn a 'good' feature representation for the target domain. In this case, the knowledge used to transfer across domains is encoded into the learned feature representation. With the new feature representation, the performance of the target task is expected to improve significantly. (Baralis et al., 2008) present a method for learning a low-dimensional representation which is shared across a set of multiple related tasks. Based on the well-known 1-norm regularization problem which provides such a sparse representation for a single task case, they generalize this formulation to a multiple task case. Their method learns a few features common across the tasks by regularizing within the tasks while keeping them coupled to each other. Moreover, the method can be used, as a special case, to select a few features from a prescribed set. Since the extended problem is non-convex, they develop an equivalent convex optimization problem and present an algorithm to solve it. The learning algorithm simultaneously learns both the features and the task functions through two alternating steps. The first step consists of independently learning the parameters of the tasks' regression or classification functions. The second step consists of learning, in an unsupervised way, a low-dimensional representation for these task parameters, which they show to be equivalent to learning common features across the tasks. The number of common features learned is controlled by the regularization parameters, much like sparsity is controlled in the case of single-task 1-norm regularization.

The parameter transfer approach assumes that the source tasks and the target tasks share some parameters or the prior distribution of the hyperparameters of the models. The transferred knowledge is encoded into the shared parameters or priors as knowledge can be transferred across tasks. (Dai et al., 2007a) propose an approach to multi-task learning based on the minimization of regularized functions similar to existing ones, such as the one for SVM, which has been successfully used in the past for single-task learning. Their approach is able to model the relation between tasks in terms of a novel kernel function that uses a task-coupling parameter. This is the first generalization of regularization-based methods from single-task to multi-task learning. They implement an instance of the proposed approach similar to SVM and test it empirically using both simulated and real data. The experimental results show that the proposed method performs better than existing multi-task learning methods and largely outperforms single-task learning using SVMs.

Relational knowledge transfer learning deals with knowledge transfer for relational domains. The basic assumption behind this context is that some relationship between the data in the source and target domains are similar. Thus, the knowledge to be transferred is the relationships between the data. (Pedrycz, 2013) presents an approach to mapping source knowledge when minimal target domain data is available. In particular, their approach addresses the single-entity-centered setting in which the learner is provided with information concerning only a single entity. The single-entity-centered setting can be viewed as one extreme on the spectrum of possible available relational data. They present a new method, called the Short-Range to Long-Range, which evaluates possible source-to-target predicate correspondence based on short-range clauses in order to also transfer the knowledge captured in long-range clauses. The experiments additionally show that the most accurate model for the source domain is not always the best model to use for transfer.

2.3 Types of transfer learning models based on domain knowledge

In addition, models to transfer learning can be divided into three types according to how much knowledge exists in each domain. 1) Supervised knowledge transfer aims to only use the labeled instances in the source domains to transfer knowledge to a target domain (target domain has some labeled instances (Duan et al., 2012). 2) Semi-supervised knowledge transfer addresses transfer problems when there are unlabeled instances in the source domains that could be used to help improve accuracy in the target domain (the target domain has some labeled instances (Li et al., 2014). 3) Unsupervised knowledge transfer transfers knowledge from a related domain to an unknown target domain that does not have any labeled instances (Gong et al., 2014). For each type of transfer learning model, transfer learning feature spaces can be either homogeneous feature spaces, where the source and target domain have the same features, or heterogeneous feature spaces, where the source and target domain have different features.

In the supervised knowledge transfer, one representative model that deals with the domain adaptation in homogeneous spaces is the method of transfer component analysis (Pan et al., 2011). It measures the distance between two domains using **m**aximum **m**ean **d**iscrepancy (MMD), and minimizes this distance to make the source domain and target domain become closer to each other. Another model is the geodesic flow kernels model (Gong et al., 2014), which integrates the subspaces between the source and target domain using the geodesic flow of a Grassmann manifold. (Shi and Sha, 2012) proposed a method of information-theoretical learning, which learns the common feature space for two domains, and the specific feature space for each domain separately. Based on the extracted feature spaces, the information-theoretic metric is optimized to improve the accuracy of the target model. The models that handle the heterogeneous domain adaptation include manifold alignment-based models (Wang and Mahadevan, 2011), asymmetric regularized cross-domain transformation (Kulis et al., 2011), and heterogeneous spectral mapping (Shi et al., 2013).

In the semi-supervised knowledge transfer, many methods have been proposed to solve the distribution divergence between two domains in the homogeneous space. (Kanamori et al., 2009) treat the domain adaptation problem as the covariate shift, and increase the weights of the source data that are located in the dense region in the target domain to approximate the distribution of the target domain.

However, this method does not work well when there is a big gap between the source and target domains or some important target features can be represented in the source domain. Self-labeling adaptation models (Jiang and Zhai, 2007) adopts a self-training method to update the source model with labeled target data. Also, the performance highly depends on the initial prediction model, and is not suitable in cases where there is a huge divergence between the source and target domains. For the heterogeneous domain adaptation problem, DASH-N (Nguyen et al., 2015) is proposed to extract the hierarchy of features together with the transformations that rectify the mismatch between the source and target domains. This method also has good performance in relation to object recognition.

In unsupervised knowledge transfer, we assume that there are no labeled data available in the target domain, and all the data in the source domain are labeled. Homogeneous unsupervised knowledge transfer is studied as a part of homogeneous supervised knowledge transfer when there are no labeled data available in the target domain. The representative models include transfer component analysis (Pan et al., 2011), geodesic flow kernels (Gong et al., 2014), and information-theoretic metric (Wang and Mahadevan, 2011). The heterogeneous unsupervised knowledge transfer is a very challenging issue and only a few research studies have attempted to solve it. Kernel canonical correlation analysis (Yeh et al., 2014) was proposed to address this issue, but there is a strict condition that the data in the two domains must be paired.

2.4 Types of transfer learning based on computational intelligence techniques

There are several types of transfer learning models based on computational intelligence techniques, including NN, Bayes, fuzzy system, and genetic algorithms. These different types of transfer learning models are reviewed in this section.

2.4.1 Transfer learning using NN

NN aims to solve complex non-linear problems using a learning-based method inspired by the human brain structure and processes. In classical machine learning problems, many studies have demonstrated the superior performance of NN compared to statistical methods. This has encouraged many researchers to use NN for transfer learning, particularly for complicated problems. To address the problem in transfer learning, a number of NN-based transfer learning algorithms have been developed in recent years. This section reviews three of the principal NN techniques: Deep NN, Multiple Tasks NN, and Radial Basis Function NN, and presents their application to transfer learning.

2.4.1.1 Transfer learning using deep NN

Deep NN is considered to be an intelligent feature extraction module that offers great flexibility in extracting high-level features in transfer learning. The prominent characteristic of deep NN is its multiple hidden layers, which can capture the intricate non-linear representations of data. (Hubel and Wiesel, 1962) proposed multi-stage Hubel-Wiesel architectures that consist of alternating layers of convolutions and max pooling to extract data features. A new model blending the above structure and multiple tasks is proposed for transfer learning (Ahmed et al., 2008). In this model, a target task and related tasks are trained together with shared input and hidden layers, and separately output neurons. The model is then extended to the case in which each task has multiple output neurons (Huang et al., 2013). Likewise, based on the multi-stage Hubel-Wiesel architectures, whether shared hidden layers trained by the source task can be reused on a different target task is detected. For the target task model, only the last classification layer needs to be retrained, but any layer of the new model could be fine-tuned if desired. In this case, the parameters of hidden layers in the source task model act as initialization parameters of the new target task model, and this strategy is especially promising for a model in which good initialization is very important (Cireşan et al., 2012). As previously mentioned, generally all the layers except the last layer are treated as feature extractors in a deep NN. In contrast to this network structure, a new feature extractor structure is proposed by (Collobert and Weston, 2008). Only the first two layers are used to extract features at different levels, such as word level and sentence level in **n**atural **l**anguage **p**rocessing (NLP). Subsequent layers are classical NN layers used for prediction. The **s**tacked **d**enoising **a**utoencoder (SDA) is another structure that is presented in deep NN [50]. The core idea of SDA is that unsupervised learning is used to pre-train each layer, and ultimately all layers are fine-tuned in a supervised learning way. Based on the SDA model, different feature transference strategies are introduced to target tasks with varying degrees of complexity. The number of layers transferred to the new model depends on the high-level or low-level feature representations that are needed. This means if low-level features are needed, only the first layer parameters are transferred to the target task (Kandaswamy et al., 2014). In addition, an interpolating path is presented to transfer knowledge from the source task to the target task in a deep NN. The original high dimensional features of the source and target domains are projected to lower dimensional subspaces that lie on the Grassman manifold, which presents a way to interpolate smoothly between the source and target domains; thus, a series of feature sets is generated on the interpolating path and intermediate feature extractors are formed based on deep NN (Chopra et al., 2013). Deep NN can also combine with other technology to promote the performance of transfer learning. (Swietojanski et al., 2012) applied **r**estricted **B**oltzmann **m**achine (RBM) to pre-train deep NN, and the outputs of the network are used as features for a hidden Markov model.

2.4.1.2 Transfer learning using multiple task NN

To improve the learning for the target task, multiple task learning (MTL) is proposed. Information contained in other related tasks is used to promote the performance of the target task (Caruna, 1993). In multiple task NN learning, all tasks are trained in parallel using the shared input and hidden neurons and separate output neurons depending on different tasks (Caruana, 1998). The biggest difference between the MTL and the MTL in deep NN is the number of hidden layers. Generally, the number of hidden layers in MTL is much smaller than in deep NNs. In MTL, source tasks as auxiliary information help the target task to improve performance. However, due to different relatedness between source tasks and the target task, the contributions of source tasks should be distinguished. Therefore, a modified version of multitask learning called ηMTL is introduced. Based on a measure of relatedness between source tasks and the target task, ηMTL applies a separate learning rate for each task output neuron (Silver and Mercer, 2001). (Silver and Mercer, 2002) presented a task rehearsal method (TRM) to transfer knowledge of source tasks to the target task at a functional level. Instead of the interrelation between representations of various tasks, the relationship between functions of tasks is the core content in their new model. After demonstrating the good performance of ηMTL and TRM on synthetic tasks, they were practically applied to a series of medical diagnostic tasks (Silver and Mercer, 2007). In the MTL model, the output layer consists of a separate neuron corresponding to each task, which may lead to redundant outputs and overlapping information. In addition, for the continuous tasks, contextual cues must be provided to guide the system to associate an example with a particular task. In light of these problems, (Silver and Poirier, 2007) proposed context-sensitive multiple task learning (csMTL) with two major differences. To eliminate the redundant outputs and reduce the free parameters, only one neuron is included in the output layer. The other difference is reflected in the input layer, which can be divided into two parts. Apart from the set of input variables for the tasks, the input layer also contains a set of context inputs that associates each training example with a particular task. To verify the effectiveness of csMTL, a set of experiments was designed to detect csMTL and MTL NNs in their ability to transfer knowledge (Silver et al., 2008). The above model makes the assumption that each task only has one output neuron. Further, csMTL is extended to learn tasks that have multiple output neurons (Silver and Tu, 2008).

2.4.1.3 Transfer learning using radial basis function NNs

(Yamauchi, 2008) considered covariate shift, one category of transfer learning, and incremental learning. Under the assumption that incremental learning environments are a subset of covariate shift, a novel incremental learning method is presented on the basis of radial basis function NN. Further, a number of model-selection criteria are set up to optimize the network; for example, the information criterion (Shimodaira, 2000) is applied to determine the number of hidden neurons (Yamauchi, 2009). In some of the literature, NN acts as a part of the whole algorithm. (Liu et al., 2009) applied NN to initialize the weights of labeled data in the source domain. Each instance in the source domain is input into the NN trained by limited target labeled data to obtain the contribution degree based on the error value. In addition, the NN is used as a pre-processing technique to extract features from high dimensional space to low dimensional space (Ueki et al., 2010). Sometimes, NN is combined with other intelligent techniques to form an integrated model to improve the performance of transfer learning (Celiberto Jr et al., 2011).

2.4.2 Transfer learning using Bayes

Bayesian techniques are methods that are related to statistical inference and are developed based on Bayesian theory. A Bayesian classifier is a probabilistic methodology for solving classification problems. Since probability is a useful tool for modeling the uncertainty in the real world and is adequate for quantifying the certainty degree of an uncertain truth, a Bayesian classifier is popular in the machine learning community. When it comes to the transfer learning setting, the distribution of the training data and test data is not identical, so a Bayesian classifier trained on training data may not be predictive for the test data. To address this challenging problem, many Bayesian-based transfer learning algorithms have been developed in recent years. This section reviews three of the main Bayesian techniques: naïve Bayes classifier, Bayesian network and the hierarchical Bayesian model, and illustrates their application within the framework of transfer learning.

2.4.2.1 Transfer learning using naïve Bayes

The naïve Bayes classifiers Lewis (1992) are among the most popular classifiers in real-world applications. They pose a simple but strong assumption that there is independence between each pair of features (attributes) given the class variables. Though this assumption is not suitable in most real scenarios, naïve Bayes classifiers have nevertheless been proved to work quite well in some complicated applications, especially automatic medical diagnosis (Kononenko, 1993), spam filtering (Androutsopoulos et al., 2000) and text categorization (Sebastiani, 2002), in which they may even outperform more advanced algorithms, such as support vector machine, or random forests.

To adapt the naïve Bayes classifier from the training data to the test data, (Dai et al., 2007b) proposed a novel naïve Bayes transfer learning (NBTL) classification algorithm for text categorization. NBTL first trains a naïve Bayes classifier on the training data and applies the learned classifier on the test data to obtain a pseudo label for the test data during learning, thereby providing an initial model estimation for the test data under the target distribution. The **e**xpectation **m**aximization (EM) algorithm is then applied in the iteration to find a local optimal model only for fitting the target distribution, meaning that the naïve Bayes classifier trained on the training data is adapted to the test data. To measure the difference between the different distributions, KL divergence is used to estimate a trade-off parameter in the NBTL model, and the experiment results show that the performance of NBTL increases when the distribution between the training data and the test data is significantly different. The main disadvantage of NBTL lies in the fact that the influence of new-domain specific features is ignored. Instead of treating both old-domain and new-domain data equally, an adaptive naïve Bayes is proposed in (Tan et al., 2009). It uses a weighted EM algorithm to dynamically increase the importance of new-domain data and decrease the weight of old data, while at the same time emphasizing the usage of both generalizable features drawn from the old-domain data and all the features from the new-domain data for tackling the cross-domain sentiment classification problem. (Roy and Kaelbling, 2007) developed an alternative method of transferring the naïve Bayes classifier. They first partition the dataset into a number of clusters, such that the data for each cluster for all tasks has the same distribution. Then they train one classifier for each partition; all classifiers are then combined using a Dirichlet process.

In addition to text classification, (Ma et al., 2012) developed a transfer naïve Bayes (TNB) algorithm to predict cross-company software defects. The implementation can be summarized in three steps: it first collects maximum and minimum value vectors of the target feature from test data, then each feature of a training sample is compared with the corresponding part of those two vectors to calculate the number of similar attributes and the weight of that training instance is computed through a gravitational analogy. After obtaining all the weights for the training data, a prediction model can be built with those weighted training data to classify the test dataset.

2.4.2.2 Transfer learning using Bayesian network

Assuming that total independence between features is not applicable for many real-world problems, the occurrence of an event may arise as the result of a number of relevant factors. In other words, there are correlations between features in a decision region and the Bayesian network is a suitable representation to this fact. A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest. It consists of two components: (1) a directed acyclic graph (DAG) which contains nodes and arcs. In particular, the nodes can be observed quantities, latent variables, or unknown parameters, while the directed arcs reflect conditional dependencies among variables, and (2) conditional probability tables (CPTs), which record local probability distributions associated with each node. Bayesian networks have four distinct advantages when compared to other data mining methods, namely, the ability to handle incomplete datasets, to discover causal relationships hidden in the data, to incorporate both domain knowledge and data into one model, and to avoid data over-fitting (Heckerman et al., 1998).

In a simple case, the graphical model of a Bayesian network can be constructed by the prior knowledge of an expert. However in some complex applications, the definition of "network" is difficult for humans, so it is necessary to learn the network structure and parameters of the local distributions from data (Buntine, 1996). To learn a Bayesian network from data, one needs to consider two important phases: structure learning and parameter learning, respectively. The former relates to the learning of a graphical model from data, while the latter deals with the evaluation of condition probability distribution for each variable given the model. To our knowledge, most work focuses on structure learning by leveraging previous data, and less effort is expended on parameter learning.

When the training data in a task is scarce, learning a reliable Bayesian network is difficult; therefore transfer learning can help improve the robustness of learned networks by exploiting data from related tasks or knowledge from domain experts. In (Niculescu-Mizil and Caruana, 2007), the authors extended Bayesian network learning from a single domain (task) to multiple domains (tasks). In this case, instead of learning a structure in isolation, the relationships between tasks should be taken into account. Similar to the multi-task learning scenario, multiple Bayesian network structures are jointly built from multiple related datasets. To make learning efficient, the parameters of Bayesian networks from related tasks are assumed to be independent. The prior is defined in such a way that it penalizes structures that are different from one another. A score and search approach is then performed on the space of multiple Bayesian networks to find the best structures, in particular, by defining a suitable search space and devising a branch and bound procedure that enables efficient moves in this search space. In contrast to learning optimal models simultaneously for different tasks, (Luis et al., 2010) proposed learning models from auxiliary tasks to improve related tasks. In this paper, without providing sufficient data for an independence test, a PC-TL algorithm is developed with consideration of both the confidence of the independence tests and the similarity of the auxiliary tasks to the target task in a combined function. An example that uses transfer learning to strengthen the quality of the learned Bayesian networks through the use of an inductive bias can also be found in (Richardson and Domingos, 2003). The main limitation of such multi-task network structure learning algorithms lies in the assumption that all pairs of tasks are equally related, which violates the truth that different pairs of tasks can differ in their degree of relatedness. As a result, (Oyen et al., 2012) relaxed this assumption by adding a task relatedness metric, which explicitly controls the amount of information sharing between tasks, into a learning objective to incorporate domain knowledge about task-relatedness. Experimental results show that leveraging domain knowledge produces models that are both robust and in accordance with a domain expert's objective. Recently, (Oyen and Lane, 2013) pointed out that it is more appropriate to estimate a posterior distribution over multiple learned Bayesian networks rather than a single posteriori. In their paper, the authors proposed the incorporation of structure bias into order-conditional network discovery algorithms to extend network discovery in individual Bayesian network learning (Friedman and Koller, 2003; Koivisto and Sood, 2004) for transfer network learning.,

Given a Bayesian network structure, the work of parameter learning is to estimate the conditional probability tables (CPTs) for each node given the combination of its parent's nodes. If we have data from all tasks, then we can directly estimate the CPTs from the data. However, in some cases, we only have models from related tasks and need to estimate the CPT for the target task. In (Luis et al., 2010), two novel aggregation methods were defined. The first calculates a weighted average of the probabilities from the data of the auxiliary tasks based on the confidence of the probability estimated from the auxiliary tasks and the similarity with the target estimates. This average is then combined with the target probability estimate, weighted by a factor that depends on its similarity to the target probability. The second method works similarly, but the average of probabilities is obtained from those closer to the target rather than from all the data of the auxiliary tasks. In addition, the average is combined to the target estimate with a confidence factor, which is based on the amount of data.

2.4.2.3 Transfer learning using a hierarchical Bayesian model

Hierarchical Bayesian models are considered to be a particular type of Bayesian network and are used when the data are structured in groups. This hierarchical model can represent and reason about knowledge at multiple levels of abstraction, therefore a hierarchical Bayesian model provides a unified framework to explain both how abstract knowledge is used for induction and how abstract knowledge can be acquired.

In considering the problem of multi-task learning, (Wilson et al., 2007) used a hierarchical Bayesian infinite mixture model to model the distribution over multiple **M**arkov **d**ecision **p**rocesses (MDP) such that the characteristics of new environments can be quickly inferred based on the learned distribution as an informed prior. This idea is extended to solve the problem of sequential decision-making tasks (Wilson et al., 2012). (Yang et al., 2008) combined all the tasks in a single RBF network and defined a novel Bayesian multi-task learning model for non-linear regression. (Raykar et al., 2008) presented a novel Bayesian **m**ultiple **i**nstance **l**earning (MIL) algorithm which performs feature selection and classifier construction simultaneously. The results show that the proposed method is more accurate and effective when a smaller set of useful features is selected.

In reference to the domain adaptation problem, a novel hierarchical Bayesian domain adaptation model was developed based on the use of a hierarchical Bayes prior (Finkel and Manning, 2009). In the proposed model, several parameters are set to each feature in each domain, and the top level parameters are proposed on the upper level such that the Gaussian prior over the parameter values in each domain are now centered around these top level parameters instead of around zero, while the zero-mean Gaussian prior is placed over the top level parameters. At the same time, (Wood and Teh, 2009) designed a doubly hierarchical Pitman-Yor process language model, in which the bottom layer utilizes multiple hierarchical Pitman-Yor process language models to represent a number of domains while the top layer is responsible for sharing the statistical strength. A more special case is considered in (Salakhutdinov et al., 2012), where only a single example from a new category is provided; thus, it is more difficult to estimate the variance and similarity metric for categorizing an object in this case. It is possible with this model to encode priors for new classes into super-categories. Following the inference of the sub-category to which the novel category belongs, the model can estimate not only the mean of the new category but also an appropriate similarity metric based on parameters inherited from the super-category.

2.4.3 Transfer learning using fuzzy systems and genetic algorithms

Imprecision, approximation, vagueness and ambiguity of information are driven by the variability encountered when trying to learn an activity with little information. There is a clear co-dependency on the level of certainty in any learning activity and the amount of information that is available, and problems with little information, can have a high degree of uncertainty. This is why several research studies appears very recently applied fuzzy techniques to transfer learning. The use of fuzzy logic allows for the incorporation of approximation and a greater expressiveness of the uncertainty within the knowledge transfer. (Zadeh, 1965) introduced the concept of fuzzy sets on which he later expanded on by introducing further aspects of fuzzy logic, including fuzzy rules in (Bellman and Zadeh, 1970). The two primary elements within fuzzy logic, the linguistic variable and the fuzzy if-then rule, are able to mimic the human ability to capture imprecision and uncertainty within knowledge transfer. Fuzzy logic forms a major component of the published fuzzy transfer learning techniques. (Behbood et al., 2011, 2013a) developed a fuzzy-based transductive transfer learning for long-term bank failure prediction in which the distribution of data in the source domain differs from that in the target domain. They applied three classical predictors, NN, SVM and fuzzy NN, to predict the initial labels for samples in the target domain, then attempted to refine the labels using fuzzy similarity measures. The authors subsequently improved the performance of the fuzzy refinement domain adaptation method (Behbood et al., 2014) by developing a novel fuzzy measure to simultaneously take account of the similarity and dissimilarity in the refinement process. The proposed method has been applied to text categorization and bank failure prediction. The experiment results demonstrated the superior performance of the proposed method compared to popular classical transductive transfer learning methods. Using fuzzy techniques in similarity measurement and label production, the authors demonstrated the advantage of fuzzy logic in knowledge transfer where the target domain lacks critical information and involves uncertainty and vagueness. (Shell and Coupland, 2012, 2015) proposed a framework of fuzzy transfer learning to form a prediction model in intelligent environments. To address the issues of modeling environments in the presence of uncertainty and noise, they introduced a fuzzy logic-based transfer learning that enables the absorption of the inherent uncertainty and dynamic nature of transfer knowledge in intelligent environments. They created a transferable fuzzy inference system using labeled data in the source domain, then adapted and applied the resultant rule base to predict the labels for samples in the target domain. The source rules were adjusted and adapted to the target domain using the Euclidean distance measure. The proposed method was examined in two simulated intelligent environments. The experiment results demonstrated the superior performance of fuzzy transfer learning compared to classical prediction models; however the method has not been compared with transfer learning methods. (Deng et al., 2014) proposed the generalized hidden-mapping ridge regression (GHRR) method to train various types of classical intelligence models, including NNs, fuzzy logic systems and kernel methods. The knowledge-leverage based transfer learning mechanism is integrated with GHRR to realize the inductive transfer learning method called transfer GHRR (TGHRR). Since the information from the induced knowledge is much clearer and more concise than the information from the data in the source domain, it is more convenient to control and balance the similarity and difference of data distributions between the source and target domains. The proposed GHRR and TGHRR algorithms have been evaluated experimentally by performing regression and classification on synthetic and real world datasets. The results demonstrated that the performance of TGHRR is competitive with or even superior to existing state-of-the-art inductive transfer learning algorithms.

Genetic algorithms are an evolutionary method that simulates the process of natural selection to mainly solve optimization and search problems. This method uses techniques inspired by natural evolution such as inheritance, mutation, selection and crossover. (Koçer and Arslan, 2010) introduced the use of genetic algorithm and transfer learning by extending a previously constructed algorithm. Their approach was to extend the transfer learning method of producing a translation function. This process allows for differing value functions that have been learn to be mapped from source to target tasks. The authors incorporated the use of a set of policies originally constructed by a genetic algorithm to form the initial population for training the target task. They showed that the transfer of inter-task mappings can reduce the time required to learn a second, more complex task.

2.4.4 Applications of transfer learning

Transfer learning approaches with the support of computational intelligence methods such as NN, Bayesian networks, and fuzzy logic have been applied in real-world applications. These applications largely fall into the following five categories: 1) NLP; 2) computer vision; 3) biology; 4) finance; and 5) business management.

2.4.4.1 Natural language processing

NLP, which can be regarded as the study of human languages, is proposed to make natural language processing interpretable by computers. In general, there are numerous sub-learning tasks in NLP fields, such as text-based learning problems (e.g., text classification or non-topical text analysis), language knowledge understanding, etc.

For text-related analysis, i.e., exploring useful information from a given document, the learning problem of how to label text documents across different distributions was addressed (Dai et al., 2007b). In this setting, the labeled training samples shared different distributions from the unlabeled test data. Accordingly, a novel transfer-learning algorithm based on an EM based Naive Bayes model was proposed for further learning, which has demonstrated the best performance on three different types of data sets. Moreover, considering that most existing transfer learning methods assume that features and labels are numeric, and lack the ability to handle the uncertainty properties, (Behbood et al., 2013b) proposed a fuzzy domain adaptation approach and carried out an investigation into its applicability to text classification. In addition, for sentiment classification, which is a key challenge in non-topical text analysis, the transfer learning technique is also applicable, such as adapting naïve Bayes to domain adaptation for sentiment analysis by fully utilizing the information from both the old-domain and unlabeled new-domain data sets (Tan et al., 2009).

Furthermore, the transfer learning approach can be used to deal with language knowledge understanding problems. For speech recognition, for example, (Swietojanski et al., 2012) exploited untranscribed acoustic data into the target languages in a deep NN on unsupervised cross-lingual knowledge transfer. Similarly, (Huang et al., 2013) dealt with the cross-language knowledge transfer learning tasks by a shared-hidden-layer multi-lingual deep NN.

2.4.4.2 Computer vision and image processing

Computer vision applied to transfer learning using computational intelligence includes methods for acquiring, processing, analysing, and understanding images, especially in high-dimensional data from the real world, to produce numerical or symbolic information. In this section, we summarize computer visual applications for camera image processing, from digits to letter processing, and video processing.

In early camera image applications based on computational intelligent transfer learning, all approaches used a database of camera images of different objects, each of which was of a distinct color or size and was used for vision learning, such as ALVINN-like road-following vision recognition (Caruana, 1998). One challenge in image object recognition is that the distributions of the training images and test images are different. Thus, (Chopra et al., 2013) argued that in the representation learning camp for images, existing deep learning approaches could not encode the distributional shift between the source and target domains. To this end, the authors proposed a novel transfer deep learning method for object recognition which allows the application of deep learning for domain adaptation. Camera images were also used to solve robotics problems. A visual object tracking routine, which recognizes and tracks the marker in real time, challenged robot researchers (Thrun, 1994, 1996) and a robot-mounted camera (Caruana, 1998) was employed for task mappings, e.g. (Taylor et al., 2007). Recently, image learning has mainly been used for human facial recognition, e.g., gender and ethnicity recognition based on facial appearance (Ahmed et al., 2008), emotional facial appearance recognition derived from synthetic images of neutral faces to that of corresponding images of angry, happy and sad faces (Silver and Tu, 2008), age estimations from face images (Ueki et al., 2010), and gaze gesture recognition by eye tracking devices and eye gaze technologies (Shell and Coupland, 2012). Knowledge transfer between different handwritten character recognition tasks (Cireşan et al., 2012) is another kind of application of transfer learning in computer vision. (Kandaswamy et al., 2014) trained a NN to classify Latin digits (specific source problem) and reused it to classify lowercase letters (different but related target problem) without having to train it from scratch. In the empirical analysis, the authors used the proposed NN to transfer knowledge from Arabic digits to Latin digits as well. (Kandaswamy et al., 2014) also considered a problem of classifying images of English lowercase a-to-z by reusing fine-tuned features of English handwritten digits 0-to-9.

By applying salient feature detection and tracking in videos to simulate fixations and smooth pursuit in human vision, (Zou et al., 2012) successfully implemented an unsupervised learning algorithm in a self-taught learning setting. With concrete recognition, features learned from natural videos do not only apply to still images, but also give competitive results on a number of object recognition benchmarks.

2.4.4.3 Biology

Transfer learning has been applied to biology fields, including medical problems, biological modeling designs, ecology issues, and so on. In applications related to medical issues, (Caruna, 1993) suggested using multi-task learning in artificial NNs, and proposed an inductive transfer learning approach for pneumonia risk prediction. A life-long inductive learning approach (Silver and Mercer, 2002) retained task knowledge in a representational form and transferred knowledge in another form of virtual examples on three heart disease domains, through an NN-based multi-task learning algorithm. They also put forward another type of sequential inductive transfer model for a medical diagnostics task, i.e., coronary artery disease diagnosis (Silver and Mercer, 2007). Recently, (Oyen and Lane, 2013) argued that existing transfer learning methods for Bayesian networks focus on a single posteriori estimation, and that in doing so, other models may be ignored. To this end, they proposed a transfer multi-Bayesian networks model for whole-brain neuroimaging.

From the aspect of biological modeling designs, e.g., robot bionics, (Celiberto Jr et al., 2011) combined three artificial intelligence techniques, case-based reasoning, heuristically accelerated reinforcement learning and NNs, in a transfer learning problem. They then proposed a novel model called L3 to speed up the reinforcement learning framework for a set of empirical evaluations between two domains (the Acrobot and the Robocup 3D). Another important biology domain, ecology, has attracted the attention of researchers into transfer learning. For instance, (Niculescu-Mizil and Caruana, 2007) proposed a multi-task Bayesian network structure learning (i.e., inductive transfer) to re-evaluate the performance of ALARM (a logical alarm reduction mechanism) and to handle a real bird ecology problem in North America.

2.4.4.4 Finance

Another application area of transfer learning is finance, such as in the area of car insurance risk estimations and financial early warning systems. (Niculescu-Mizil and Caruana, 2007) presented an inductive transfer learning approach, which jointly learns multiple Bayesian network structures instead of adaptive probabilistic networks from multiple related data sets. The authors examined the proposed method using car insurance risk estimation networks. It is worth noting that the works on intelligent financial warning systems and long-term prediction in banking ecosystems (Behbood et al., 2014) are the first systematic studies to apply transfer learning approaches using fuzzy logic techniques of computational intelligence to real-world financial applications to exploit the knowledge of the banking system, e.g., transferring information from one country to establish a prediction model in another country.

2.4.4.5 Business management

Transfer learning using computational intelligence has been applied in business management. For instance, (Roy and Kaelbling, 2007) proposed an efficient Bayesian task-level transfer learning model to tackle the user's behavior in the meeting domain. (Jin and Sun, 2008) indicated that traditional NN methods for traffic flow forecasting are based on a single task which cannot utilize information from other tasks. To address this challenge, multi-task based NN is proposed to transfer knowledge to deal with traffic flow forecasting. Luis, (Luis et al., 2010) proposed the use of a novel transfer Bayesian network learning framework, including structure and parameter learning, to handle a product manufacture process issue. Recently, (Ma et al., 2012) studied the cross-company software defect prediction scenario in which the source and target data sets come from different companies, and proposed a novel transfer naïve Bayes as the solution. A dynamic model for intelligent environments has been proposed to make use of the data from different feature spaces and domains (Shell and Coupland, 2012), with a novel fuzzy transfer learning process.

2.5 Granular computing

2.5.1 Concepts of granular computing

Information granules as intuitively appealing constructs play a pivotal role in human cognitive and decision-making activities (Pedrycz and Chen, 2015). We perceive complex phenomena by organizing existing knowledge along with available experimental evidence. Knowledge is structured in a form of some meaningful, semantically sound entities, which are central to all ensuing processes of describing the world, reasoning about the environment and support decision-making activities.

In general, an information granule is regarded as a collection of elements drawn together by their closeness (resemblance, proximity, functionality, etc.) articulated in terms of some useful spatial, temporal, or functional relationships (Pedrycz et al., 2013). Granular computing involves representing, constructing and processing information granularity. Information granules permeate almost all human endeavors. No matter which problem is taken into consideration, we usually set it up in a certain conceptual framework composed of some generic and conceptually meaningful entities - information granules, which we regard to be of relevance to the problem formulation, further problem solving, and a way in which the findings are communicated to the community. Information granules realize a framework in which we formulate generic concepts by adopting a certain level of abstraction.

Information granules are the key components of knowledge representation and processing. The level of granularity of information granules (their size, to be more descriptive) becomes crucial to the problem description and an overall strategy of problem solving, and the hierarchy of information granules supports an important aspect of perception of phenomena and delivers a tangible way of dealing with the complexity of the system by focusing on the most essential facets of the problem. There is no universal level of granularity of information; commonly the size of granules is problem-oriented and user dependent (Pedrycz, 2001). Human-centricity is an inherent feature of intelligent systems. It is anticipated that two-way effective human-machine communication is imperative. Humans perceive the world, reason, and communicate at some level of abstraction. Abstraction comes hand-in-hand with non-numeric constructs, which embrace collections of entities characterized by some notions of closeness, proximity, resemblance, or similarity.

In the algorithmic realization of granular computing, the implicit nature of information granules has to be translated into constructs that are explicit in nature, viz. described formally in which information granules can be efficiently processed. It identifies the essential commonalities between the surprisingly diversified problems and technologies used there, which could be cast into a unified framework known as a granular world. A fully operational processing entity interacts with the external world (that could be another granular or numeric world) by collecting necessary granular information and returning the outcomes of the granular computing (Pedrycz and Homenda, 2013).

2.5.2 The formalisms of granular computing

The existing plethora of formalisms of granular computing includes set theory (interval analysis), rough sets, fuzzy sets, shadowed sets, probability, random sets and so on. These existing well-established development are placed under the same roof by clearly visualizing that in spite of their visibly distinct underpinnings (and ensuing processing), they exhibit some fundamental commonalities. In this sense, granular computing establishes a highly stimulating environment of synergy between the individual approaches. By building upon the commonalities of the existing formal approaches, granular computing helps assemble heterogeneous and multifaceted models of processing of information granules by clearly recognizing the orthogonal nature of some of the existing and well-established frameworks (Pedrycz and Homenda, 2013).

Granular computing fully acknowledges a notion of variable granularity whose range could cover detailed numeric entities and very abstract and general information granules. It looks at the aspects of compatibility of such information granules and the ensuing communication mechanisms of the granular worlds. Information granules arise as an evident realization of the fundamental paradigm of abstraction. Next, we introduce the existing platforms separately. Sets (intervals) realize a concept of abstraction by introducing a notion of dichotomy. We allow an element to belong to a given information granule or to be excluded from it. Along with set theory comes a well-developed discipline of interval analysis. Sets are described by characteristic functions taking on values 0 or 1. Fuzzy sets are important conceptual and algorithmic generalization of sets. By admitting partial membership of an element to a given information granule, we bring an important feature to ensure the concept has rapport with reality. It helps working with the notions where the principle of dichotomy is neither justified nor advantageous (Pedrycz and Gomide, 2007). Shadowed sets are the description of information granules by distinguishing those elements which fully belong to the concept, those which are excluded from it, and those whose belongingness is completely unknown (Pedrycz and Vukovich, 2002). Formally, these information granules are described as a mapping, where the elements with membership quantified as the entire [0,1] interval are used to describe a shadow of the construct. Probability-oriented information granules are expressed in the form of some probability density functions or probability functions. They capture a collection of elements resulting from some experiment. In virtue of the concept of probability, the granularity of information becomes a manifestation of the occurrence of some elements. For instance, each element of a set comes with a probability density function truncated to [0,1], which quantifies a degree of membership to the information granule. Rough sets emphasize a roughness of description of a given concept X when being realized in terms of the indiscernibility relation provided in advance. The roughness of the description of X is manifested in terms of its lower and upper approximations associated with a certain rough set (Lin, 1998).

Next, we see the information granules from a symbolic and numeric view. A concept - information granule is viewed as a single symbol (entity). This view is very much present in the AI community, where computing revolves around symbolic processing. In addition, information granules are associated with a detailed numeric characterization. Fuzzy sets are profound examples in this regard. We start with numeric membership functions. All ensuing processing involves numeric membership grades so, in essence, it focuses on number crunching.

2.5.3 The application of granular computing

In the image processing area, in spite of the continuous progress in the area, a human being assumes a dominant and very much uncontested position when it comes to understanding and interpreting images. We do not focus our attention on individual pixels and process them as such but group them together into semantically meaningful constructs – familiar objects we deal with in everyday life. Such objects involve regions that consist of pixels or categories of pixels drawn together because of their proximity in the image, similar texture, color, etc (Butenkov, 2004). This remarkable and unchallenged ability of humans is due to our effortless ability to construct information granules, manipulate them and arrive at sound conclusions.

In the processing and interpretation of time series, from our perspective we can describe time series in a semi-qualitative manner by pointing at specific regions of such signals. One distinguishes some segments of temporal signals and interprets their combinations. For example, in the stock market, one analyzes numerous time series by looking at existing amplitudes, trends, patterns, and the relationships among them. Time is another important and omnipresent variable that is subjected to granulation. We use seconds, minutes, days, months, and years. Depending on the specific problem we have in mind, who the user is and the size of the information granules (time intervals) could vary quite significantly (Chen and Chen, 2015).

Another application is the design of software systems. We develop software artifacts by admitting a modular structure of an overall architecture of the designed system. Each module is a result of identifying essential functional closeness of some components of the overall system. Modularity (granularity) is the holy grail of the systematic software design, supporting the production of high quality software products (Han and Dong, 2007).

2.6 Summary

Transfer learning has become a hot topic in machine learning and an powerful tool for a model's construction in a new area which does not have too much data. This chapter has reviewed the current state-of-the-art researches in this field. The different types of transfer learning methods, these being content transfer, domain knowledge and computational intelligence techniques, are reviewed. The reviews have shown that transfer learning is experiencing its developing stage, so there is still a gap between current theories and techniques with complicated real-world tasks, especially complicated heterogeneous domain adaptation problems, where the different feature space is quite a gap between the two domains. Therefore, this study aims to fill this gap by contributing some new theories and techniques to this field, and applies these new theories and techniques on complex transfer learning tasks.

Chapter 3

Fuzzy Homogeneous Domain Adaptation

3.1 Introduction

In the homogeneous domain adaptation problems, the source data and target data have the same feature space but different distributions. The discrepancy of the distributions in two domains makes the source model have a poor accuracy when solving the tasks in the target domain. Each feature in the feature space is assumed to be determined by some hidden features. For example, the credit risk, an index of a bank, is determined by loan exposure, loan funding, past due loan rate, loss rate, and so on. In two domains, for instance the banks in America and the banks in Australia, they both have the index of the credit risk as a feature, but the decisive factors of this index are not identical in the two countries, which leads to the same feature with different distributions.

There have been a significant amount of research that studies transfer learning for classification problems, whereas studies on regression problems are still scarce. In this chapter, we focus on addressing regression domain adaptation problems using regression transfer learning techniques. Given that fuzzy system modeling is an important category of modeling with extensive applications (Lemos et al., 2011), incorporating regression transfer learning to a fuzzy model holds promise. Additionally, domains which lack information tend to suffer from uncertainty, and fuzzy regression transfer learning can cope efficiently with uncertainty. In transfer learning, target tasks in new environments often exhibit this uncertainty, especially when there is insufficient information, therefore a fuzzy system combined with transfer learning might exhibit a substantial capacity to model uncertainty.

A novel method of changing the input space is proposed to make the existing fuzzy model of the source domain fit the regression tasks in the target domain. The idea behind changing the input variables is that each input variable is assumed to be determined by some hidden features, and the distributions of the input variables in two domains are different because of different hidden features or different weights of these features. The approach of changing the input space adjusts the hidden features and their corresponding weights, and target data are used to complete this process so that the modified input distribution is much more similar with the target domain.

The main contributions of this chapter are twofold. First, a novel approach to regression, based on the Takagi-Sugeno fuzzy model, is introduced to address situations in which insufficient training data is available in the target domain but there is sufficient training data in the source domain. The Takagi-Sugeno model's fuzzy rules are constructed using source data, then modified through mappings to be reused to estimate values in the target domain. Second, this approach preserves the privacy of the source data because only the fuzzy rules are extracted.

The rest of this chapter is organized as follows. Section 3.2 introduces the Takagi-Sugeno fuzzy model, which is the basic model applied in our domain adaptation method. Section 3.3 discusses the problem we aim to solve and presents a fuzzy rule-based method to deal with the domain adaptation problem in homogeneous space. The input space is modified by mappings so that the existing fuzzy rules of the source domain can be suitable for the prediction tasks in the target domain. Section 3.4 explains the evaluation and analysis of the experimental results for the proposed approach. Finally, summary of this chapter is discussed in Section 3.5.

3.2 Takagi-Sugeno fuzzy model

This section discusses the Takagi-Sugeno fuzzy model, which is the basic model applied in our work to construct fuzzy rules to solve the prediction tasks. The details of the construction procedures for Takagi-Sugeno fuzzy model are given with more details in the following.

The Takagi-Sugeno fuzzy model is an effective way to represent a fuzzy model in a nonlinear dynamic system. A Takagi-Sugeno model, composed of c fuzzy rules, is formally represented as:

$$\text{if } \boldsymbol{x} \text{ is } A_i(\boldsymbol{x},\ \boldsymbol{v}_i),\ \text{then } y \text{ is } L_i(\boldsymbol{x}, \boldsymbol{a}_i) \qquad i = 1, 2, ..., c \tag{3.1}$$

Each fuzzy rule comprises one condition, which is described by the prototype $\boldsymbol{v}_i$, and one conclusion, which is typically governed by the coefficients of the linear function L_i of the input variables $\boldsymbol{a}_i$. When the input of the Takagi-Sugeno fuzzy model is $\boldsymbol{x}$, the output y is represented as:

$$y = \sum_{i=1}^{c} A_i(\boldsymbol{x}, \boldsymbol{v}_i) L_i(\boldsymbol{x}, \boldsymbol{a}_i) \tag{3.2}$$

The construction of this fuzzy rule-based model uses a set of instances $\{(\boldsymbol{x}_1,\ y_1), ..., (\boldsymbol{x}_N,\ y_N)\}$ to formulate condition A_i and optimize the parameters of the linear function L_i. The design procedure can be summarized in two steps (Hadjili and Wertz, 2002):

Step 1: Forming the conditions $A_1, A_2, ..., A_c$ through fuzzy clustering.

Typically, **f**uzzy **C**- **m**eans (FCM) (Havens et al., 2012) is used to construct the clusters and calculate the centers of clusters $\boldsymbol{v}_i$. FCM partitions the N data $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$ into c clusters, where $1 < c < N$. As a result, a collection of c centers of clusters, $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_c$, and a partition matrix, $U = [u_{ik}], i = 1, 2, ..., c, k = 1, 2, ..., N$ are formed. The partition matrix satisfies two requirements $u_{ik} \in [0, 1], \sum_{i=1}^{c} u_{ik} = 1, \forall k$ and $0 < \sum_{k=1}^{N} u_{ik} < N, \forall i$.

The objective function 3.3 is minimized in the FCM:

$$J = \sum_{i=1}^{c} \sum_{k=1}^{N} (u_{ik})^m \|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2 \tag{3.3}$$

where $\|\cdot\|$ stands for a distance function, and m $(m < 1)$ is a fuzzification coefficient that affects the shape and overlap among the resulting membership functions.

Since real-world data have variables located in different ranges, a weighted Euclidean distance avoids bias towards any particular variable. The distance is expressed in the form

$$\|\boldsymbol{x}_k - \boldsymbol{v}_i\|^2 = \sum_{j=1}^{N} \frac{(x_{kj} - v_{ij})^2}{\sigma_j^2} \tag{3.4}$$

where σ_j is the standard deviation of the jth feature, and n is the dimensionality of the input data.

The centers of clusters are calculated as:

$$\boldsymbol{v}_i = \frac{\sum_{k=1}^{N} (u_{ik})^m \boldsymbol{x}_k}{\sum_{k=1}^{N} (u_{ik})^m} \tag{3.5}$$

and the entries of the partition matrix are expressed as follows:

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \frac{\|\boldsymbol{x}_k - \boldsymbol{v}_i\|}{\|\boldsymbol{x}_k - \boldsymbol{v}_j\|}^{\frac{2}{m-1}}} \tag{3.6}$$

The entire process is repeated until no significant changes to the entries of the partition matrix U are reported in successive iterations of the algorithm.

Step 1 results in c centers of clusters, $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_c$, that determine $A_1, A_2, ..., A_c$, and $A_i(\boldsymbol{x},\ \boldsymbol{v}_i)$ can be calculated in the form:

$$A_i(\boldsymbol{x}_k,\ \boldsymbol{v}_i) = \frac{1}{\sum_{j=1}^{c} \frac{\|\boldsymbol{x}_k - \boldsymbol{v}_i\|}{\|\boldsymbol{x}_k - \boldsymbol{v}_j\|}^{\frac{2}{m-1}}} \tag{3.7}$$

Step 2: Optimizing the parameters of the linear function $L_i(\boldsymbol{x}, \boldsymbol{a}_i)$.

Suppose we have a pair of data $(\boldsymbol{x}, y)$. When the input of the Takagi-Sugeno fuzzy model is $\boldsymbol{x}$, the output is denoted as $\hat{y}$ and determined as:

$$\hat{y} = \sum_{i=1}^{c} A_i(\boldsymbol{x},\ \boldsymbol{v}_i) L_i(\boldsymbol{x}, \boldsymbol{a}_i) \tag{3.8}$$

where $L_i(\boldsymbol{x}, \boldsymbol{a}_i) = \boldsymbol{a}_i^T \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix}$, $\boldsymbol{a}_i$ is the coefficients vector of linear function L_i in the form of $\boldsymbol{a}_i = \begin{pmatrix} a_{i0} & a_{i1} & \dots & a_{in} \end{pmatrix}^T$, and $\boldsymbol{x}$ is n-dimensional input data, $\boldsymbol{x} = \begin{pmatrix} x_1 & x_2 & \dots & x_N \end{pmatrix}^T$.

Next, we simplify 3.8:

$$\hat{y} = \sum_{i=1}^{c} A_i(\boldsymbol{x},\ \boldsymbol{v}_i) \boldsymbol{a}_i^T \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix} = \sum_{i=1}^{c} \boldsymbol{a}_i^T \begin{pmatrix} A_i(\boldsymbol{x},\ \boldsymbol{v}_i) \\ A_i(\boldsymbol{x},\ \boldsymbol{v}_i)\boldsymbol{x} \end{pmatrix} \tag{3.9}$$

We denote $\boldsymbol{z}_i(\boldsymbol{x}) = \begin{pmatrix} A_i(\boldsymbol{x},\ \boldsymbol{v}_i) \\ A_i(\boldsymbol{x},\ \boldsymbol{v}_i)\boldsymbol{x} \end{pmatrix}$, then $\hat{y}$ can be rewritten as:

$$\hat{y} = \sum_{i=1}^{c} \boldsymbol{a}_i^T \boldsymbol{z}_i(\boldsymbol{x}) = \sum_{i=1}^{c} \boldsymbol{z}_i^T(\boldsymbol{x})\boldsymbol{a}_i = \boldsymbol{f}^T(\boldsymbol{x})\boldsymbol{a} \tag{3.10}$$

where $\boldsymbol{f}^T(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{z}_1(\boldsymbol{x}) & \boldsymbol{z}_2(\boldsymbol{x}) & \dots & \boldsymbol{z}_c(\boldsymbol{x}) \end{pmatrix}, \boldsymbol{a} = \begin{pmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \dots & \boldsymbol{a}_n \end{pmatrix}^T$.

Therefore we find $\hat{y} = \boldsymbol{f}^T(\boldsymbol{x})\boldsymbol{a}$, i.e. for the given input $\boldsymbol{x}$, the output of the Takagi-Sugeno fuzzy model is parameter $\boldsymbol{a}'$s linear function. Additionally we expect that $\hat{y}$ will approximate y, which is the target value corresponding to $\boldsymbol{x}$. This can be achieved by an appropriate parameter $\boldsymbol{a}$ through a calculating process based on the dataset $\boldsymbol{G} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_N, y_N)\}$.

For all the inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$, the corresponding outputs are calculated as:

$$\hat{y}_i = \boldsymbol{f}^T(\boldsymbol{x}_i)\boldsymbol{a} \qquad i = 1, 2, ..., N \tag{3.11}$$

We combine all the outputs and denote this as $\hat{\boldsymbol{Y}}$:

$$\hat{\boldsymbol{Y}} = \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}^T(\boldsymbol{x}_1)\boldsymbol{a} \\ \boldsymbol{f}^T(\boldsymbol{x}_2)\boldsymbol{a} \\ \dots \\ \boldsymbol{f}^T(\boldsymbol{x}_N)\boldsymbol{a} \end{pmatrix} = \boldsymbol{F}\boldsymbol{a} \tag{3.12}$$

where $\boldsymbol{F} = \begin{pmatrix} \boldsymbol{f}^T(\boldsymbol{x}_1) \\ \boldsymbol{f}^T(\boldsymbol{x}_2) \\ \dots \\ \boldsymbol{f}^T(\boldsymbol{x}_N) \end{pmatrix} = \begin{pmatrix} \boldsymbol{z}_1^T(\boldsymbol{x}_1) & \boldsymbol{z}_2^T(\boldsymbol{x}_1) & \dots & \boldsymbol{z}_c^T(\boldsymbol{x}_1) \\ \boldsymbol{z}_1^T(\boldsymbol{x}_2) & \boldsymbol{z}_2^T(\boldsymbol{x}_2) & \dots & \boldsymbol{z}_c^T(\boldsymbol{x}_2) \\ \dots & \dots & \dots & \dots \\ \boldsymbol{z}_1^T(\boldsymbol{x}_N) & \boldsymbol{z}_2^T(\boldsymbol{x}_N) & \dots & \boldsymbol{z}_c^T(\boldsymbol{x}_N) \end{pmatrix}$.

$\hat{\boldsymbol{Y}}$ is expected to be as close as $\boldsymbol{Y} = \begin{pmatrix} y_1 & y_2 & \dots & y_N \end{pmatrix}^T$, which is the target value corresponding to inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$.

Our aim is to minimize the objective function as follows:

$$Q = (\boldsymbol{F}\boldsymbol{a} - \boldsymbol{Y})^T(\boldsymbol{F}\boldsymbol{a} - \boldsymbol{Y}) \tag{3.13}$$

Since Q is a quadratic function of $\boldsymbol{a}$, the optimal $\boldsymbol{a}$ can be obtained analytically:

$$\boldsymbol{a}_{opt} = (\boldsymbol{F}^T\boldsymbol{F})^{-1}\boldsymbol{F}^T\boldsymbol{Y} \tag{3.14}$$

Therefore, the fuzzy rule's conclusion can be calculated based on the derived $\boldsymbol{a}$:

$$L_i(\boldsymbol{x}_k, \boldsymbol{a}_i) = a_{i0} + a_{i1}x_{k1} + \cdots + a_{in}x_{kn} \quad i = 1, \ldots, c \tag{3.15}$$

Based on steps 1 and 2, the conditions and conclusions of the fuzzy rules are formed, and a Takagi-Sugeno fuzzy model is built.

3.3 Fuzzy domain adaptation

3.3.1 Problem statement

The dataset in the source domain is denoted by $\boldsymbol{D} = \{(\boldsymbol{x}_1^s, y_1^s), (\boldsymbol{x}_2^s, y_2^s), ..., (\boldsymbol{x}_{N_s}^s, y_{N_s}^s)\}$, where $\boldsymbol{x}_k^s \in R^n, k = 1, 2, ..., N_s$ is the n-dimensional input variable, the label $y_k^s \in R$ is the continuous output variable, and N_s indicates the number of data. Since the amount of source data with labels is massive, a well-performing regression model for the source domain can be learned.

The dataset in the target domain contains two subsets: one with labels and one without labels $\boldsymbol{H} = \{(\boldsymbol{x}_1^t, y_1^t), ..., (\boldsymbol{x}_{N_t}^t, y_{N_t}^t)\}$, where $\boldsymbol{x}_k^t \in R^n, k = 1, 2, ..., N_t$ is the n-dimensional input variable, $y_k^t \in R, k = 1, 2, ..., N_{t1}$ is the continuous output variable. The number of data in $\boldsymbol{H}$ is much less than the data in $\boldsymbol{D}$, i.e. $N_t << N_s$, and not sufficient to build a good model.

Although data in the source domain and target domain have the same dimension of the input space, the marginal probability distribution of the input variables in two domains are quite different. The model of source domain, therefore, could not be directly used to solve the tasks in the target domain. Since we suppose that the source domain and target domain have some common knowledge and relate to each other, the knowledge in the existing model of source domain could be used to help the construction of target model.

3.3.2 Fuzzy rule-based domain adaptation in homogeneous space

We propose a fuzzy rule-based method to use the accumulated knowledge (fuzzy rules) in the source domains to assist the solution of the target tasks. The main idea of our fuzzy transfer learning method is presented in this subsection, and the procedures required to implement it are described in subsection 3.3.3.

A Takagi-Sugeno fuzzy model M^s for the source domain can be built based on the dataset $\boldsymbol{D}$.
model M^s

$$\text{if } \boldsymbol{x}^s \text{ is } A_i(\boldsymbol{x}^s,\ \boldsymbol{v}_i^s),\ \text{then } y^s \text{ is } L_i(\boldsymbol{x}^s, \boldsymbol{a}_i^s) \qquad i = 1, 2, ..., c \tag{3.16}$$

For the target domain, suppose the ideal model is M^t
model M^t

$$\text{if } \boldsymbol{x}^t \text{ is } A_i(\boldsymbol{x}^t,\ \boldsymbol{v}_i^t),\ \text{then } y^t \text{ is } L_i(\boldsymbol{x}^t, \boldsymbol{a}_i^t) \qquad i = 1, 2, ..., c \tag{3.17}$$

Here, we suppose the numbers of fuzzy rules are the same in both source and target domains.

Building a well-performing Takagi-Sugeno fuzzy model needs a large amount of data with labels, and inadequately data in $\boldsymbol{H}_L$ cannot guarantee performance of the constructed model in the target domain. Furthermore, discrepancies between the source and target data mean that using the source model to solve target tasks is impossible.

Given that there is insufficient data to train a new fuzzy model M^t, we hope to use learned knowledge (fuzzy rules) in the existing model M^s to help construct a fuzzy model that is more compatible with the target data. This requires the optimization of a continuous mapping of each input variable in the input space of the target data. The input space is transformed to $\Phi(\boldsymbol{x}^t)$ by mapping Φ, and a new fuzzy model $\overline{M}^t$ is constructed using the fuzzy rules from fuzzy model M^s.

Model $\overline{M}^t$, described in the forms of fuzzy rules, is:

$$\text{if } \boldsymbol{x}^t \text{ is } A_i(\Phi(\boldsymbol{x}^t),\ \Phi(\boldsymbol{v}_i^s)),\ \text{then } y^t \text{ is } L_i(\Phi(\boldsymbol{x}^t), \boldsymbol{a}_i^s) \qquad i = 1, 2, ..., c \tag{3.18}$$

The output of model $\overline{M}^t$ is calculated as:

$$g^t = \sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}^t),\ \Phi(\boldsymbol{v}_i^s)) L_i(\Phi(\boldsymbol{x}^t), \boldsymbol{a}_i^s) \tag{3.19}$$

Our aim is to find such Φ so that $\overline{M}^t \approx M^t$, i.e.

$$\sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}^t),\ \Phi(\boldsymbol{v}_i^s)) L_i(\Phi(\boldsymbol{x}^t), \boldsymbol{a}_i^s) \approx y^t \tag{3.20}$$

The key to our fuzzy regression transfer learning method is to map the input space through Φ. A nonlinear continuous function based on sigmoid functions is used to construct the mapping Φ. Other forms of mapping suitable for the problem could also be considered.

3.3.3 Knowledge transfer in fuzzy rule-based models

The procedure for transferring knowledge from a source domain to a target domain requires two steps. First, a Takagi-Sugeno fuzzy model, based on source data, is constructed, then a new fuzzy model for the target domain is built by modifying the input space using fuzzy rules from the source domain.

Step 1: Constructing a Takagi-Sugeno fuzzy model M^s based on source data.

Based on the dataset $\boldsymbol{D}$, a Takagi-Sugeno fuzzy model M^s for the source domain is constructed.

Model M^s, described in the forms of fuzzy rules, is:

$$\text{if } \boldsymbol{x}_k^s \text{ is } A_i(\boldsymbol{x}_k^s,\ \boldsymbol{v}_i^s),\ \text{then } y^s \text{ is } L_i(\boldsymbol{x}_k^s, \boldsymbol{a}_i^s) \qquad i = 1, 2, ..., c \tag{3.21}$$

The main blocks of a fuzzy rule are the condition and conclusion, which are dominated by the prototype and linear function respectively. The fuzzy model M^s is constructed by calculating the centers of clusters of the data and estimating the parameters of the linear functions standing in the conclusions of the rules.

1) Forming the prototypes

Fuzzy C-means (FCM) is used to cluster the input data $\{\boldsymbol{x}_k\}$ and find the prototypes:

$$\boldsymbol{V}^s = [\boldsymbol{v}_1^s\ \boldsymbol{v}_2^s \cdots \boldsymbol{v}_c^s]^T \tag{3.22}$$

where c is the number of clusters, i.e. the number of fuzzy rules.

Estimating an “optimal” number of clusters in the clustering algorithm is also still an open issue, and the solution to this problem greatly depends upon the data and the problem. Here, we adopt a strategy of dynamically changing the number of clusters (centers of clusters). The number of clusters (c) is chosen from a certain range and a fuzzy regression transfer learning model is constructed for each selected c. The model with the best transfer performance is then chosen to address the problem at hand.

The input data $\boldsymbol{x}_k^s$ therefore belongs to the prototype (cluster) $\boldsymbol{v}_i^s$ with the membership degree $A_i(\boldsymbol{x}_k^s, \boldsymbol{v}_i^s)$, which is calculated as follows:

$$A_i(\boldsymbol{x}_k^s, \boldsymbol{v}_i^s) = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|\boldsymbol{x}_k^s - \boldsymbol{v}_i^s\|}{\|\boldsymbol{x}_k^s - \boldsymbol{v}_j^s\|} \right)^{\frac{2}{m-1}}} \tag{3.23}$$

2) Developing linear functions

Since the functions in the conclusion are linear, they are uniquely described by the coefficients $\boldsymbol{a}_i^s$, $i = 1, 2, \ldots, c$. Based on the analysis completed in Section 3.2, the coefficients of the linear functions are calculated as follows:

$$[\boldsymbol{a}_1^s\ \boldsymbol{a}_2^s \cdots \boldsymbol{a}_c^s] = (\boldsymbol{F}^T\boldsymbol{F})^{-1}\boldsymbol{F}^T\boldsymbol{Y} \tag{3.24}$$

where $\boldsymbol{F} = \begin{pmatrix} \boldsymbol{f}^T(\boldsymbol{x}_1^s) & \ldots & \boldsymbol{f}^T(\boldsymbol{x}_N^s) \end{pmatrix}^T$, $\boldsymbol{f}^T(\boldsymbol{x}_k^s) = \begin{pmatrix} A_1(\boldsymbol{x}_k^s, \boldsymbol{v}_1^s) & \ldots & A_c(\boldsymbol{x}_k^s, \boldsymbol{v}_c^s) \\ A_1(\boldsymbol{x}_k^s, \boldsymbol{v}_1^s)\boldsymbol{x}_k^s & \ldots & A_c(\boldsymbol{x}_k^s, \boldsymbol{v}_c^s)\boldsymbol{x}_k^s \end{pmatrix}$.

$[\boldsymbol{a}_1^s\ \boldsymbol{a}_2^s \cdots \boldsymbol{a}_c^s]$ is the coefficient matrix of the linear functions, where $\boldsymbol{a}_i = [a_{i0}^s\ a_{i1}^s \cdots a_{in}^s]^T$ is the coefficient vector of the ith linear function L_i, $i = 1, 2, \ldots, c$.

As a consequence, the centers of clusters and linear functions are calculated to construct the fuzzy rules in 3.21. When a new datum $\boldsymbol{x}_k^s$ appears, the output of fuzzy model M^s is calculated as follows:

$$g_k^s = \sum_{i=1}^{c} A_i(\boldsymbol{x}_k^s,\ \boldsymbol{v}_i^s) L_i(\boldsymbol{x}_k^s, \boldsymbol{a}_i^s) \tag{3.25}$$

where $A_i(\boldsymbol{x}_k^s, \boldsymbol{v}_i^s) = 1/\sum_{j=1}^{c}\left(\frac{\|\boldsymbol{x}_k^s - \boldsymbol{v}_i^s\|}{\|\boldsymbol{x}_k^s - \boldsymbol{v}_j^s\|}\right)^{\frac{2}{m-1}}$, $L_i(\boldsymbol{x}_k^s, \boldsymbol{a}_i^s) = a_{i0}^s + a_{i1}^s x_1^s + \ldots + a_{in}^s x_n^s$.

This fuzzy model M^s will not perform well on the target data $\boldsymbol{H}$, however. To improve its performance, the input space of the target domain must be made more compatible with M^s's fuzzy rules.

Step 2: Modifying the input space and using the fuzzy rules of the existing model. Nonlinear mappings based on sigmoid functions are use to modify the input space of target domain. The mappings' construction and optimization are presented as follows.

The nonlinear function is constructed through a network that is composed of P nodes in the hidden layer and a single node at the output layer. The transformation of the jth input variable of data $\boldsymbol{x}_k^t$ is shown in Fig. 3.1 as an example of the nonlinear mapping for each input variable.

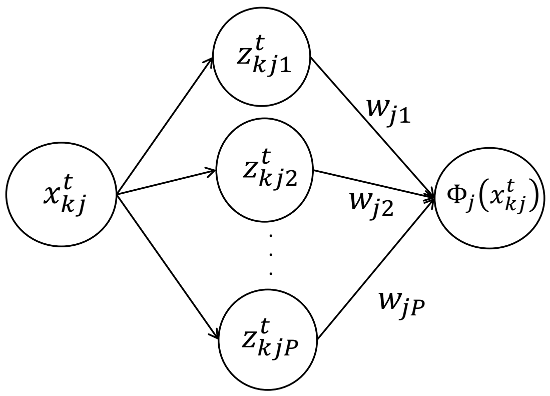


Figure 3.1 Architecture of nonlinear mapping

The active functions of the nodes in the hidden layer are sigmoid functions, which are dominated by two parameters. Therefore, as shown in Fig. 3.1, the graphic representation of the transformed jth input variable of data $\boldsymbol{x}_k^t$ is:

$$\Phi_j(\boldsymbol{x}_{kj}^t) = \sum_{p=1}^{P} w_{jp} * z_{kj}^t \tag{3.26}$$

where $z_{kj}^t = \frac{1}{1+e^{-\alpha_{jp}(x_{kj}^t - \beta_{ip})}}, j = 1,2,...,n, p = 1,2,...P,\ \alpha_{jp}$. w_{jp} indicates the weights of the pth node's contribution to the output, and satisfies $\sum_{p=1}^{P} w_{jp} = max\, x_j^t = max\{x_{1j}^t, x_{2j}^t, ..., x_{Nj}^t\}$

By taking advantage of the nonlinear mappings, the input space is transformed so that the new input variables become more compatible with the fuzzy rules of the existing fuzzy model.

The input space is modified by constructing a mapping for each input variable, say $\Phi_j(x^t_{kj})$, where x^t_{kj} is the jth input variable of $\boldsymbol{x}^t_k$. The input space is thus transformed by $\Phi = [\Phi_1\Phi_2\cdots\Phi_n]$, and the input data $\boldsymbol{x}^t_k$ becomes $\Phi(\boldsymbol{x}^t_k)$. The specific form of $\Phi(\boldsymbol{x}^t_k)$ depends on the construction method for Φ. As described above, nonlinear continuous mappings, shown in Fig. 3.1, are used here, and the corresponding $\Phi(\boldsymbol{x}^t_k)$ is calculated through 3.27:

$$\Phi(\boldsymbol{x}^t_k) = \begin{pmatrix} \Phi_1(\boldsymbol{x}^t_{k1}) \\ \Phi_2(\boldsymbol{x}^t_{k2}) \\ \cdots \\ \Phi_n(\boldsymbol{x}^t_{kn}) \end{pmatrix} = \begin{pmatrix} \sum_{p=1}^{P} w_{1p}/(1+e^{-\alpha_{1p}(\boldsymbol{x}^t_{k1}-\beta_{1p})}) \\ \sum_{p=1}^{P} w_{2p}/(1+e^{-\alpha_{2p}(\boldsymbol{x}^t_{k2}-\beta_{2p})}) \\ \cdots \\ \sum_{p=1}^{P} w_{np}/(1+e^{-\alpha_{np}(\boldsymbol{x}^t_{kn}-\beta_{np})}) \end{pmatrix} \tag{3.27}$$

Take advantage of the mappings, nonlinear functions, transformations are made to the input spaces so that the new input variables become more compatible to the fuzzy rules of the existing fuzzy model.

Since Φ transforms the whole input space, both the input data of the dataset $\{\boldsymbol{x}^t_k\}$ and the centers of clusters in 3.22 need to be transformed using the mapping Φ. The prototype $\boldsymbol{v}^s_i$ becomes $\Phi(\boldsymbol{v}^s_i)$. Model M^s's fuzzy rules are transferred to the new data space, and a new model $\overline{M}^t$ is built.

model $\overline{M}^t$

$$\text{if } \boldsymbol{x}^t_k \text{ is } A_i(\Phi(\boldsymbol{x}^t_k),\ \Phi(\boldsymbol{v}^s_i)),\ \text{then } y^t_k \text{ is } L_i(\Phi(\boldsymbol{x}^t_k), \boldsymbol{a}^s_i) \qquad i = 1,2,\cdots,c \tag{3.28}$$

where $A_i(\Phi(\boldsymbol{x}^t_k), \Phi(\boldsymbol{v}^s_i)) = 1/\sum_{j=1}^{c}(\frac{\|\Phi(\boldsymbol{x}^t_k)-\Phi(\boldsymbol{v}^s_i)\|}{\|\Phi(\boldsymbol{x}^t_k)-\Phi(\boldsymbol{v}^s_j)\|})^{\frac{2}{m-1}}$,

$L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s) = a_{i0}^s + a_{i1}^s \Phi_1(x_{k1}^t) + \ldots + a_{in}^s \Phi_n(x_{kn}^t)$, $i = 1, 2, \ldots, c$.

Therefore, when the input is $\boldsymbol{x}_k^t$, the output of model $\overline{M}^t$ is:

$$g_k^t = \sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}_k^t), \Phi(\boldsymbol{v}_i^s)) L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s) \tag{3.29}$$

For all the input data $\boldsymbol{x}_k^t$ in dataset, we find the output corresponding to 3.29, and hope the output could approximate the real output of $\boldsymbol{x}_k^t$: y_k^t. Such mapping Φ is found through the optimization of cost function in 3.30:

$$Q_t = \frac{1}{N_{t1}} \sum_{k=1}^{N_{t1}} \sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}_k^t),\ \Phi(\boldsymbol{v}_i^s)) L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s) - y_k^t)^2 \tag{3.30}$$

The parameters of Φ are optimized by minimizing 3.30. In the literature, PSO and DE are reported to be two of the most suitable global optimization algorithms (Das et al., 2008). In this chapter, we apply PSO and DE to optimize the parameters of Φ, and their performance in the fuzzy regression transfer learning is compared in detail in the section of experiments.

3.4 Empirical results analysis

To evaluate the proposed fuzzy regression transfer learning method and its learning algorithm, both synthetic and real-world datasets are adopted to validate the new method and explore the impact of optimization algorithms in the new method.

The focus of this chapter is on using the new fuzzy regression transfer technology to address regression problems. Data outputs in regression problems are more complicated than outputs in classification problems, and as a result of the fuzzy model we use, the method of generating the synthetic data is crucial and must be reasonable. Therefore, prior to demonstrating the results of the experiments, we will describe the process of generating the source data and target data and the models' construction process in Section 3.4.1. A number of symbolic representations of the experimental results are denoted in this section so that the presentation of the experiment results is clearer. Then Section 3.4.2 compares the performance of two optimization algorithms when building the models. Section 3.4.3 analyzes two different connection ways of constructing the mappings. Finally, Section 3.4.4 shows the results on some real-world datasets.

3.4.1 Generation of synthetic datasets

There are four steps in implementing our method.

Step 1: Generate source data and construct fuzzy model M^s.

Step 2: Generate target data, the number of which is much smaller than the number in the source domain, and use the existing model M^s to estimate the outputs of the target data.

Step 3: Use the data in the target domain to construct a new model $\tilde{M}^t$ for the target domain.

Step 4: Modify the input space of model M^s using the target data to obtain a new model $\overline{M}^t$ for the target domain.

We use the five-fold cross validation, which is commonly used in model validation in machine learning, to construct the models in Steps 1, 3 and 4.

More detail follows about these four steps, especially concerning the generation of the datasets.

<u>Step 1</u>: Generate source data and construct the fuzzy model M^s.

This step is divided into two sub-steps.

<u>Step 1-1</u>: Generate source data.

The process of generating source data includes the generation of the inputs and finding the corresponding outputs to constitute the input-output pairs. Referring to the input data $\boldsymbol{X}^s$, we generate three sub-datasets $\boldsymbol{X}_1^s, \boldsymbol{X}_2^s, \boldsymbol{X}_3^s$ that have different distributions, and combine them to construct $\boldsymbol{X}^s$. The parameters related to the input data are listed in Table 3.1.

Table 3.1 Parameters of input data in the source domain

Datasets	Distributions
$\boldsymbol{X}_1^s$	$x_k^s \sim N(\boldsymbol{\mu}_1^s, \boldsymbol{\sigma}_1^s),\, x_k^s \in \boldsymbol{X}_1^s, k = 1, ..., N^s/3$
$\boldsymbol{X}_2^s$	$x_k^s \sim N(\boldsymbol{\mu}_2^s, \boldsymbol{\sigma}_2^s),\, x_k^s \in \boldsymbol{X}_2^s, k = N^s/3+1, ..., 2N^s/3$
$\boldsymbol{X}_3^s$	$x_k^s \sim N(\boldsymbol{\mu}_3^s, \boldsymbol{\sigma}_3^s),\, x_k^s \in \boldsymbol{X}_3^s, k = 2N^s/3+1, ..., N^s$

We generate the input data $\boldsymbol{X}^s = \{\boldsymbol{x}_k^s\}, k = 1, ..., N_s$, where N_s is the number of source data in the source domain. Based on this, we calculate the output data in the following way. For the sake of reasonability, the outputs are generated according to 3.25, so the centers of clusters and linear functions are needed in advance. Since the datasets follow normal distributions, we assume the mean values of the normal distributions as the centers of clusters, denoted as $\overline{\boldsymbol{v}}^s$:

$$\overline{\boldsymbol{v}}^s = [\overline{\boldsymbol{v}}_1^s\ \overline{\boldsymbol{v}}_2^s\ \overline{\boldsymbol{v}}_3^s]^T = [\boldsymbol{\mu}_1^s\ \boldsymbol{\mu}_2^s\ \boldsymbol{\mu}_3^s]^T \tag{3.31}$$

The coefficients of linear functions $\overline{\boldsymbol{a}}_i^s, i = 1, 2, 3$, are given in Table 3.2.

Table 3.2 Parameters of input data in the source domain

Linear Functions	Coefficients
$L_1(\cdot, \overline{\boldsymbol{a}}_1^s)$	$\overline{\boldsymbol{a}}_1^s = [\overline{a}_{10}\ \overline{a}_{11}\ \overline{a}_{12}]$
$L_2(\cdot, \overline{\boldsymbol{a}}_2^s)$	$\overline{\boldsymbol{a}}_2^s = [\overline{a}_{20}\ \overline{a}_{21}\ \overline{a}_{22}]$
$L_3(\cdot, \overline{\boldsymbol{a}}_3^s)$	$\overline{\boldsymbol{a}}_3^s = [\overline{a}_{30}\ \overline{a}_{31}\ \overline{a}_{32}]$

Based on the centers of clusters and linear functions, when the input is $\boldsymbol{x}_k^s$, the output y_k^s can be obtained as follows:

$$y_k^s = \sum_{i=1}^{c} A_i(\boldsymbol{x}_k^s, \overline{\boldsymbol{v}}_i^s) L_i(\boldsymbol{x}_k^s, \overline{\boldsymbol{a}}_i^s) \tag{3.32}$$

We calculate the output $\boldsymbol{Y}^s = \{y_k^s\}, k = 1, \ldots, N_s$, and finally obtain the data $\boldsymbol{D} = (\boldsymbol{X}^s, \boldsymbol{Y}^s)$.

Step 1-2: Construct the fuzzy model M^s based on data $(\boldsymbol{X}^s, \boldsymbol{Y}^s)$.

The construction of a **T**akagi- **S**ugino (TS) depends on the centers of clusters $\boldsymbol{v}_i^s$ and linear functions $L_i(\cdot, \boldsymbol{a}_i^s)$. Since the construction procedure is described in Section 3.3.3, we will not go into much detail here.

We apply the five-fold cross validation when constructing model M^s, and split the dataset $\boldsymbol{D}$ into a training set $\boldsymbol{D}_1(80\%)$ and a testing set $\boldsymbol{D}_2(20\%)$. Once model M^s has been constructed, it is tested on the testing set $\boldsymbol{D}_2$, and the **m**ean **s**quare **e**rror (MSE) is denoted as:

$$Q = \frac{1}{N_{s2}} \sum_{k=1}^{N_{s2}} (y_k^t - d_k^t)^2 \tag{3.33}$$

where d_k^t is the output of model M^s when its input is $\boldsymbol{x}_k^t$, $(\boldsymbol{x}_k^t, y_k^t) \in \boldsymbol{D}_2$. N_{s2} is the number of data in the testing set $\boldsymbol{D}_2$. We calculate the mean and standard deviation of Q.

Step 2: Generate target data, the number of which is much smaller than the number in the source domain, and use the existing model M^s to estimate the outputs of the target data.

This step is also divided into two sub-steps to better describe it.

Step 2-1: Generate target data.

The method of generating input data $\boldsymbol{X}^t$ in the target domain is the same as the method used in the source domain, and the parameters related to the input data in the target domain are listed in Table 3.3.

Table 3.3 Parameters of input data in the target domain

Datasets	Distributions
$\boldsymbol{X}_1^t$	$x_k^t \sim N(\boldsymbol{\mu}_1^t, \boldsymbol{\sigma}_1^t), x_k^t \in \boldsymbol{X}_1^t, k = 1, ..., N_t/3$
$\boldsymbol{X}_2^t$	$x_k^t \sim N(\boldsymbol{\mu}_2^t, \boldsymbol{\sigma}_2^t), x_k^t \in \boldsymbol{X}_2^t, k = N^t/3+1, ..., 2N_t/3$
$\boldsymbol{X}_3^t$	$x_k^t \sim N(\boldsymbol{\mu}_3^t, \boldsymbol{\sigma}_3^t), x_k^t \in \boldsymbol{X}_3^t, k = 2N_t/3+1, ..., N_t$

The centers of clusters are also the mean values:

$$\overline{v}^t = [\overline{\boldsymbol{v}}_1^t \ \overline{\boldsymbol{v}}_2^t \ \overline{\boldsymbol{v}}_3^t]^T = [\boldsymbol{\mu}_1^t \ \boldsymbol{\mu}_2^t \ \boldsymbol{\mu}_3^t]^T \tag{3.34}$$

The coefficients of the linear functions in the target domains are given in Table 3.4.

Table 3.4 Coefficients of linear functions in the target domain

Linear Functions	Coefficients
$L_1(\cdot, \overline{\boldsymbol{a}}_1^t)$	$\overline{\boldsymbol{a}}_1^t = [\overline{a}_{10}^t \ \overline{a}_{11}^t \ \overline{a}_{12}^t]$
$L_2(\cdot, \overline{\boldsymbol{a}}_2^t)$	$\overline{\boldsymbol{a}}_2^t = [\overline{a}_{20}^t \ \overline{a}_{21}^t \ \overline{a}_{22}^t]$
$L_3(\cdot, \overline{\boldsymbol{a}}_3^t)$	$\overline{\boldsymbol{a}}_3^t = [\overline{a}_{30}^t \ \overline{a}_{31}^t \ \overline{a}_{32}^t]$

To make the source and target data different, the centers of clusters and linear functions in the source domain and target domain must also be different, i.e.

$$\overline{\boldsymbol{v}}^t \neq \overline{\boldsymbol{v}}^s, \ \overline{\boldsymbol{a}}^t \neq \overline{\boldsymbol{a}}^s \tag{3.35}$$

where $\overline{\boldsymbol{v}}^s, \overline{\boldsymbol{a}}^s, \overline{\boldsymbol{v}}^t, \overline{\boldsymbol{a}}^t$ are the centers of clusters and coefficients of the linear functions in the source domain and target domain, respectively.

Except for the centers of clusters, the covariance matrices of the target data are not the same as those of the source data. This variety in the target data is beneficial for testing the validity of our algorithm.

Based upon the input data $\boldsymbol{X}^t$, the corresponding output $\boldsymbol{Y}^t = \{y_k^t\}$ is calculated as follows:

$$y^t = \sum_{i=1}^{c} A_i(\boldsymbol{x}_k^t, \overline{\boldsymbol{v}}_i^t) L_i(\boldsymbol{x}_k^t, \overline{\boldsymbol{a}}_i^s) \tag{3.36}$$

As a consequence, we have the data $\boldsymbol{H} = (\boldsymbol{X}^t, \boldsymbol{Y}^t) = \{(\boldsymbol{x}_k^t, y_k^t)\}$ in the target domain. The number of data in $\boldsymbol{H}$ is much smaller than in $\boldsymbol{D}$ ($N^t \ll N^s$).

Step 2-2: Use the existing model M^s to estimate the outputs of the target data.

Given that the source domain and target domain have different centers of clusters and linear functions, the fuzzy model M^s does not perform well on the target data. This means that when the input is $\boldsymbol{x}_k^t$, the output h_k^t calculated using the fuzzy rules for the source data may be quite different with y_k^t. We test model M^s on target data $\boldsymbol{H}$, and the discrepancy between y_k^t and h_k^t is denoted as Q_1:

$$Q_1 = \frac{1}{N^t} \sum_{k=1}^{N'} (h_k^t - y_k^t)^2 \tag{3.37}$$

where h_k^t is the output of model M^s when the input is $\boldsymbol{x}_k^t$, $(\boldsymbol{x}_k^t, y_k^t) \in \boldsymbol{H}$, N_t is the number of data in the target domain.

Step 3: Use the data in the target domain to construct a new model $\tilde{M}^t$ for the target domain.

Proving that a model does not perform as well when trained with less data in the target domain supports our assumption. Although only a small amount of data is available in the target domain, they can nevertheless be used to train a model; the problem is that the accuracy of the model cannot be guaranteed since the training data is insufficient. The procedures for the construction of $\tilde{M}^t$ are exactly the same as are used to build model M^s for the source domain, and we also apply five-fold cross validation procedure. The target dataset $\boldsymbol{H}$ is split into a training set $\boldsymbol{H}_1$(80%) and a testing set $\boldsymbol{H}_2$(20%). Model $\tilde{M}^t$ is constructed based on the training set $\boldsymbol{H}_1$ and then used to predict the outputs for the testing set $\boldsymbol{H}_2$.

The MSE is denoted as:

$$Q_2 = \frac{1}{N_{t2}} \sum_{k=1}^{N_{t2}} (s_k^t - y_k^t)^2 \tag{3.38}$$

where s_k^t is the output of model $\tilde{M}^t$ when its input is $\boldsymbol{x}_k^t$, $(\boldsymbol{x}_k^t, y_k^t) \in \boldsymbol{H}_2$, N_{t2} is the number of data in the testing set $\boldsymbol{H}_2$. The mean and standard deviation of Q_2 are obtained.

Step 4: Modify the input space of model M^s using target data to obtain a new model $\overline{M}^t$ for the target domain.

The input space is modified by the continuous mapping Φ, which is obtained by minimizing 3.39 using the training set $\boldsymbol{H}_1$.

$$Q^{'} = \frac{1}{N_{t1}} \sum_{k=1}^{N_{t1}} (g_k^t - y_k^t)^2 \tag{3.39}$$

where g_k^t is the output of model $M^{'}$ calculated by (32) when the input is $\boldsymbol{x}_k^t$, $(\boldsymbol{x}_k^t, y_k^t) \in \boldsymbol{H}_1$, N_{t1} is the number of data in the training set $\boldsymbol{H}_1$. The optimization algorithms PSO and DE are used to find the optimal parameters of Φ.

Following the construction of model $\overline{M}^t$, it is tested on the testing set $\boldsymbol{H}_2$, and the MSE is denoted as:

$$Q_3 = \frac{1}{N_{t2}} \sum_{k=1}^{N_{t2}} (t_k^t - y_k^t)^2 \tag{3.40}$$

where t_k^t is the output of model $\overline{M}^t$ when the input is $\boldsymbol{x}_k^t$, $(\boldsymbol{x}_k^t, y_k^t) \in \boldsymbol{H}_2$, N_{t2} is the number of data in the testing set $\boldsymbol{H}_2$.

The above process also involves a five-fold cross validation procedure, so the mean and standard deviation of Q_3 are calculated.

In the following experiments, Q_1, Q_2 and Q_3 are compared, and the desired outcome is that Q_3 will be smaller than both Q_1 and Q_2. $Q_3 < Q_1$ shows that the modified model $\overline{M}^t$ is improved compared to the existing model M^s, and $Q_3 < Q_2$ demonstrates that the modified model $\overline{M}^t$ is better than model $\tilde{M}$ trained using few data in the target domain. Next, the experimental results are given to verify the effectiveness of the proposed fuzzy regression transfer learning method.

3.4.2 Comparison of evolution algorithms PSO and DE

The purpose of this experiment is to apply and compare two optimization algorithms, namely PSO and DE, and to optimize the parameters of the mappings and build the new fuzzy regression transfer learning model $\overline{M}^t$ for the target domain. Further, the validation of the new method is confirmed.

PSO and DE are computational methods that determine an optimal solution by iteratively navigating a population of solutions, which minimizes a certain predetermined objective function (performance index). There are three parameters in PSO that demonstrate significant impact on optimization performance: the inertia weight factor w, and two auxiliary parameters determining the dynamics of the population, $c1$ and $c2$. Typically, w, $c1$, and $c2$ assume value coming from several ranges, specifically $w \in [0.4, 1.2]$, $c1 \in [1.4, 2]$, $c2 \in [1.4, 2]$, whereas $c1$ is equal to $c2$ Shi and Eberhart (1998). In DE, optimization performance is largely dependent upon the values of the differential weight F and the crossover probability CR. The value range of F is $[0, 2]$, and the value range of CR is $[0, 1]$ Price et al. (2006). Based on the complexity of the problems, we apply the same initialization strategy in PSO and DE for all the experiments below; 200 candidate solutions are generated, and the maximum number of iterations is set to 200.

The distributions of the datasets applied in this experiment are shown in Table 3.5.

Table 3.5 Distributions of source data and target Data

Source Data		Target Data	
Mean values	Covariance	Mean values	Covariances
$\boldsymbol{\mu}_1^s = [1\ 1]$	$\boldsymbol{\sigma}_1^s = \begin{pmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{pmatrix}$	$\boldsymbol{\mu}_1^t = [1.5\ 1.5]$	$\boldsymbol{\sigma}_1^t = \begin{pmatrix} 0.4^2 & 0.1 \\ 0.1 & 0.4^2 \end{pmatrix}$
$\boldsymbol{\mu}_2^s = [2\ 1]$	$\boldsymbol{\sigma}_2^s = \begin{pmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{pmatrix}$	$\boldsymbol{\mu}_2^t = [2\ 1.5]$	$\boldsymbol{\sigma}_2^t = \begin{pmatrix} 0.5^2 & 0.1 \\ 0.1 & 0.5^2 \end{pmatrix}$
$\boldsymbol{\mu}_3^s = [1.5\ 2]$	$\boldsymbol{\sigma}_3^s = \begin{pmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{pmatrix}$	$\boldsymbol{\mu}_3^t = [2\ 2]$	$\boldsymbol{\sigma}_3^t = \begin{pmatrix} 0.5^2 & 0.1 \\ 0.1 & 0.5^2 \end{pmatrix}$

The linear functions in the source and target domains are shown in Table 3.6.

Table 3.6 Coefficients of linear functions in two domains

Source domain		Target domain	
$L_1(\overline{\boldsymbol{a}}_1^s)$	$\overline{\boldsymbol{a}}_1^s = [1\ 1\ 1]$	$L_1(\overline{\boldsymbol{a}}_1^t)$	$\overline{\boldsymbol{a}}_1^t = [1.5\ 0.5\ 1.5]$
$L_2(\overline{\boldsymbol{a}}_2^s)$	$\overline{\boldsymbol{a}}_2^s = [2\ 2\ 1]$	$L_2(\overline{\boldsymbol{a}}_2^t)$	$\overline{\boldsymbol{a}}_2^t = [1\ 1\ 0.5]$
$L_3(\overline{\boldsymbol{a}}_3^s)$	$\overline{\boldsymbol{a}}_3^s = [-1\ 1\ 3]$	$L_3(\overline{\boldsymbol{a}}_3^t)$	$\overline{\boldsymbol{a}}_3^t = [-1.5\ 1.5\ 4.5]$

The input of the source data and the target data is displayed in Figs. 3.2 and 3.3. As can be seen, there are some crossover regions between the source data and the target data. There are 1500 instances in the source domain, and 15 instances in the target domain.

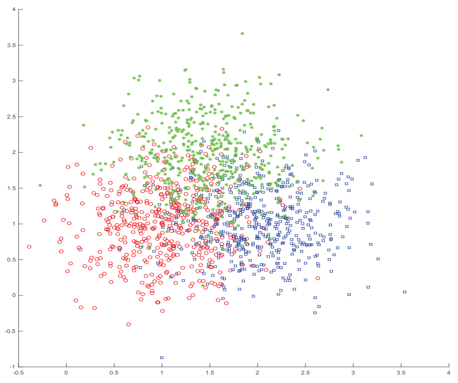


Figure 3.2 Input data of source domain

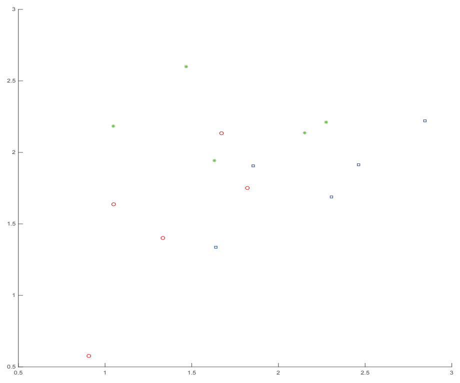


Figure 3.3 Input data of target domain

To see the difference between the source data and target data more clearly, we combine them in one Fig. 3.4. The 3-dimension points of the source data and the target data in the form of input-output pairs are drawn in Fig. 3.5.

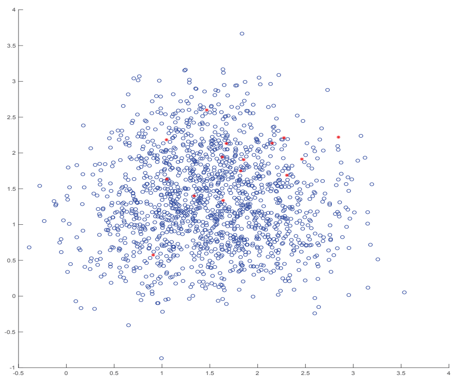


Figure 3.4 Source and target data (2D)

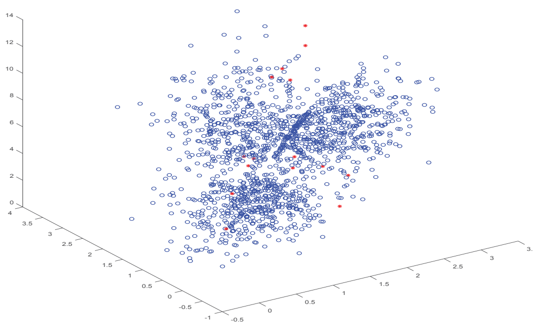


Figure 3.5 Source and target data (3D)

We see from Figs. 3.2-3.5 that the distributions of both the input and output of the source data and the target data are different.

The results of Q, Q_1 and Q_2 are shown in Table 3.7.

Table 3.7 The results of experiments

	mean $\pm$ standard deviation
Q	0.01 ± 0.00
Q_1	7.32 ± 0.12
Q_2	183.80 ± 405.14

In the process of constructing a new fuzzy model $\overline{M}^t$ for the target domain to obtain Q_3, the experiment includes two parts to analyze the stability of the PSO and DE algorithms, and the generalization of the new fuzzy regression transfer learning model.

(a) The stability of the PSO and DE algorithms is analyzed.

The stability of PSO in different w, $c1$, and $c2$ is tested, and similarly, the stability of DE in different F and CR is tested. For comparison, the same source and target datasets are applied, and the experimental results of the new fuzzy regression transfer learning model on the target domain (Q_3) are displayed in Table 3.8. Each experiment is run 10 times and we report the mean and standard deviation to quantify the stability of the solutions and the performance of the method.

Table 3.8 The stability of PSO and DE

PSO		DE	
$w, c1 = c2$	Q_3	F, CR	Q_3
0.6,1.4	2.07 ± 0.86	0.5,0.4	1.75 ± 0.15
0.6,1.7	2.24 ± 1.30	0.5,0.6	0.19 ± 0.1
0.6,2	1.84 ± 0.89	0.5,0.9	1.66 ± 0.19
0.9,1.4	1.84 ± 1.04	1,0.4	1.74 ± 0.21
0.9,1.7	2.19 ± 1.21	1,0.6	1.77 ± 0.25
0.9,2	1.71 ± 0.50	1,0.9	1.71 ± 0.05
1.2,1.4	1.66 ± 0.73	1.5,0.4	1.62 ± 0.43
1.2,1.7	2.30 ± 1.26	1.5,0.6	1.61 ± 0.32
1.2,2	2.81 ± 1.42	1.5,0.9	1.74 ± 0.37

We observe from Table 3.8 that the optimization performance of DE is better than that of PSO. Furthermore, the standard deviation in DE is smaller than the standard deviation in PSO, so the algorithmic stability of DE is superior to that of PSO.

(b) The generalization of the new fuzzy regression transfer learning model is studied.

Five-fold cross validation is applied in all the experiments. The dataset is split into five subsets, four of which are chosen as the training set, while the remaining subset forms the testing set. There are consequently five results for each model. The standard deviation of the five results indicates the generalization of the constructed model. The large standard deviation indicates either overfitting, or indicates that the data characteristics forming the training set do not coincide with the nature of the testing set. Conversely, the low value of the standard deviation shows that the model constructed on the basis of the training data has good generalization capability.

The generalization of the newly constructed model for the target domain is tested. PSO and DE with varying parameters are used to construct the model, and each experiment is run 10 times. The experimental results are shown in Table 3.9.

From the results, we observe that the values of the standard deviation, which reflects the generalization aspects of the constructed model, are not always very low. We claim that this situation is commonly encountered in transfer learning. Our assumption in the transfer learning problem is that the data in the target domain are limited in number and insufficient to develop a good model. Therefore, there is only a small set of training data with which to construct a new fuzzy regression model for the target domain, and an even smaller set of testing data. The high values of the standard deviation in the five-fold cross validation are anticipated, and clearly, these would decrease as the number of data in the target data increased. However if the size of the target data increases, the target data themselves could achieve the formation of a good model, with no need to transfer knowledge from another domain. The high values of the standard deviation in the five-fold cross validation procedure are thus reasonable and to be expected.

Table 3.9 The values of Q_3 under different parameters

PSO		DE	
$w, c1 = c2$	Q_3	F, CR	Q_3
0.6,1.4	2.39 ± 3.03	0.5,0.4	2.32 ±1.51
0.6,1.7	3.92 ± 5.91	0.5,0.6	2.14 ± 1.44
0.6,2	2.86±3.18	0.5,0.9	2.46±1.83
0.9,1.4	2.91±3.13	1,0.4	2.36±1.81
0.9,1.7	2.19±1.21	1,0.6	2.40±1.77
0.9,2	4.34±6.27	1,0.9	1.71±0.05
1.2,1.4	2.92± 3.57	1.5,0.4	2.31±1.82
1.2,1.7	3.10±3.57	1.5,0.6	2.42±0.32
1.2,2	4.77± 6.76	1.5,0.9	2.27±1.73

In all the experiments in (a) and (b), the values of Q_3 are always smaller than the values of Q_1 and Q_2, and this demonstrates that the new fuzzy regression transfer leaning model $\overline{M}^t$ is better than the existing model M^s and the model $\tilde{M}^t$ trained using few data in the target domain.

3.4.3 Comparison of different ways of constructing mappings

This experiment investigates the impact of different input space reconstruction methods when there are interactions between the features. As there are two datasets in the source domain and target domain in transfer learning problems, we consider the following four cases in Table 3.10.

Table 3.10 Distributions of source data and target data

	Source domain	Target domain
Case 1	N	N
Case 2	Y	Y
Case 3	Y	N
Case 4	N	Y

Here, "N" indicates that there is no interaction between the features, and "Y" stresses that there is interaction between the features.

In the synthetic datasets, we generate data in the following way to highlight interaction scenarios between the features. The synthetic datasets are all two-dimensional. If the functions in the conclusion part of the fuzzy rules are in the form $L = a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2$, we suppose that there is interaction between features x_1 and x_2. If we consider linear functions in the form $L = a_0 + a_1x_1 + a_2x_2$, we suppose that there is no interaction between the features.

We apply the two methods to construct the mappings for the input space with the structures shown in Fig. 3.6. In structure 1, the nonlinear mapping is constructed for each input variable. In structure 2, the nonlinear mapping is constructed for the entire input space (viz. all variables).

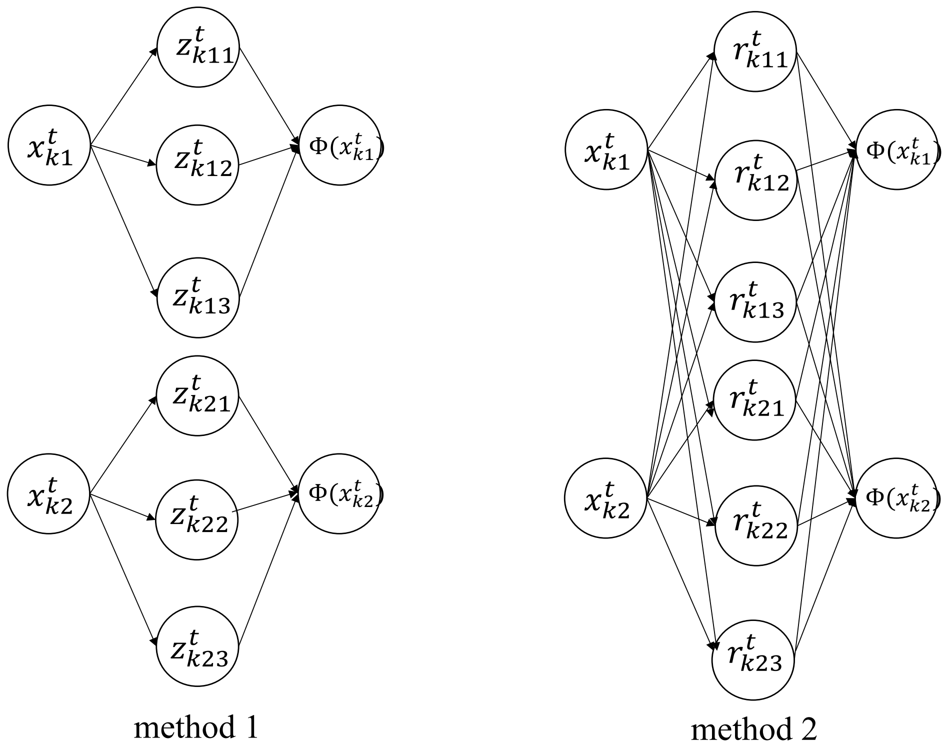


Figure 3.6 Two methods of constructing the mappings for the input space

These two methods of constructing mappings for the input space are applied to the above four cases and the results are compared to determine whether there is an interaction between the features, and which method of transforming the input space is superior. Every experiment is run 10 times, and the experimental results of the proposed fuzzy regression transfer learning model in the target domain are reported in Table 3.11.

From the results, we note that in all cases, whether or not there is an interaction between the features in the source domain and target domain, the performance of the first method is far better than the performance of the second method. Therefore, in a real-world problem where it is unknown whether there are interactions between features, it is advisable to use the first method to construct the transformation of the input space, i.e., construct the mapping for each input variable.

Table 3.11 Summary of the experimental results Q_3(mean$\pm$ standard deviation)

Cases	method 1	method 2
(N,N)	0.51±0.08	2.41±1.78
(Y,Y)	0.35±0.13	1.77±3.55
(Y,N)	0.29±0.06	2.42±4.95
(N,Y)	0.48±0.04	0.98±0.54

3.4.4 Experiments on real-world datasets

Two public datasets are applied to elaborate on the usefulness of the new fuzzy regression transfer learning in dealing with real world problems. Since the studies on regression problems of adapting a domain are scarce, there are no public datasets in these scenarios. In this work, therefore, the real-world datasets from UCI Machine Learning Repository are used and modified to simulate the regression domain adaptation problems and verify the proposed method. The way of modifying the datasets is crucial, so we give the detailed description to these datasets.

1) Housing dataset

The UCI Machine Learning Repository is a public dataset for regression problems. We use the "Housing Data Set" from this repository and revise it for the purpose of transfer learning. The dataset is split into two datasets using the attribute "TAX" to represent the full-value property-tax rate per \$10,000. Instances of "TAX" being smaller than 600 form the source dataset, and instances of "TAX" being larger than 600 constitute the target dataset. There are 360 instances in the source domain and 60 instances in the target domain. We select two attributes to construct the feature spaces (average number of rooms per dwelling, and weighted distance to five Boston employment centers), and use the attribute “MEDV”, representing the median value of owner-occupied homes in \$1000’s, as the output value. The input data in the source and target domains are shown in Fig. 3.7, and the 3-dimensional input-output data points are displayed in Fig. 3.8.

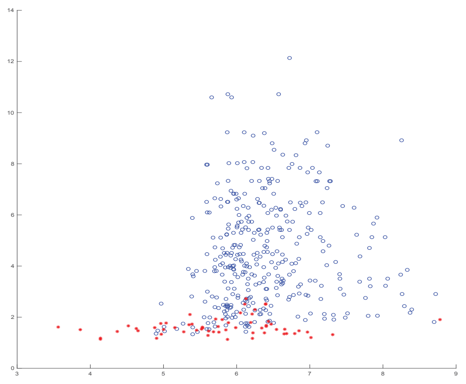


Figure 3.7 Housing data (2D)

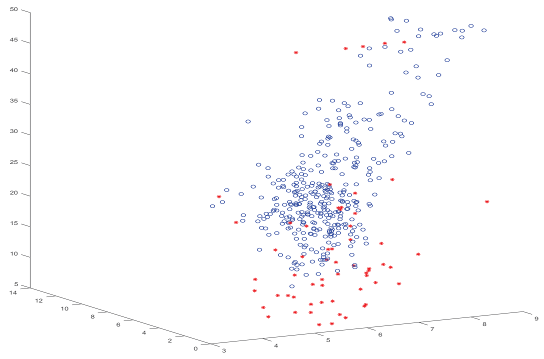


Figure 3.8 Housing data (3D)

As can be seen from Fig. 3.7 and Fig. 3.8, the distributions of both the input data and the output data in the two domains are different. The DE optimization algorithm is used to optimize the parameters, and $F = 0.5$, and $CR = 0.9$. This experiment also uses a five-fold cross validation procedure, and for the purpose of analysis, the number of clusters is fixed as 6. The results are outlined in Table 3.12 and the values of Q, Q_1, Q_2 and Q_3 are listed in Table 3.13.

Table 3.12 Results of dataset "Housing"

	mean $\pm$ standard deviation
Q	16.77 ± 9.03
Q_1	144.53 ± 13.35
Q_2	26915.57 ± 146894.50
Q_3	171.70 ± 224.09

Table 3.13 The values of Q_1, Q_2, and Q_3 for "Housing"

	Q	Q_1	Q_2	Q_3
1	16.22	140.45	94.25	70.18
2	13.79	167.55	108388.78	572.06
3	30.68	133.81	170.59	75.25
4	17.44	143.39	31.22	86.26
5	5.69	137.48	25892.99	54.75

As shown in Table 3.13, the values of Q_1 are all greater than 130, which indicates that the model for the source domain is unable to effectively solve regression tasks in the target domain. The results shown in the fourth column of the table indicate that the value of Q_2 is unstable. This is due to the small amount of available target data, so a model built from these training data cannot be generalized for the testing data.

The mean value of Q_3 shown in Table 3.12 is not small (171.70), but when analyzing the values of Q_3 in Table 3.13, we find that the other values of Q_3 are small and almost all are less than the values of Q_1 and Q_2. The results for the second experiment show an exception (572.06), but also lead to a large standard deviation. The reason for the large value of Q_3 is that the number of data in the target domain is small, so the training data does not reflect the characteristics of the testing data. Our new fuzzy regression transfer learning method therefore performs well on this real-world problem.

2) Concrete compressive strength dataset

In this series of experiments, we include another real-world dataset from the Machine Learning Repository, “Concrete Compressive Strength” data. The data has 8 attributes: “cement”, “blast furnace slag”, “fly ash”, “water,” “superplasticizer”, “coarse aggregate”, “fine aggregate”, and “age”, and the output feature is “concrete compressive strength”. We revise the dataset in two aspects to make it more appropriate for transfer learning. First, the dataset is split into a source domain and a target domain according to the attribute “age”: instances with “age” smaller than 100 fall into the source domain, and instances with “age” bigger than 100 fall into the target domain. To clearly differentiate between the source data and the target data, the attributes “blast furnace slag”, “fly ash” and “superplasticizer” are perturbed by random numbers following different distributions in two domains. There are 900 instances in the source domain and 60 instances in the target domain.

For this dataset, DE is used to optimize the parameters, and $F = 0.5$, and $CR = 0.9$. In addition, we apply the method incorporating the automatic change of number of clusters described in Section III-A, and compare the results to choose the best solution. The five-fold cross validation procedure is applied, and the results are reported in the form of *mean±standard deviation* in Table 3.14.

Table 3.14 Results of dataset "Concrete Compressive Strength"

	5 clusters	6 clusters	7 clusters	8 clusters
Q	0.02 ± 0.01	0.02 ± 0.00	0.02 ± 0.00	0.02 ± 0.00
Q_1	1.97 ± 1.22	2.65 ± 1.27	2.20 ± 0.94	2.13 ± 1.00
Q_2	$16232.88 \pm$ 259181.71	$161546.09 \pm$ 215316.00	$1240.89 \pm$ 2648.12	$249.13 \pm$ 409.39
Q_3	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00

High values of Q_1 indicate that the model in the source domain does not work well for the target data. High values of Q_2 indicate that the target data is insufficient for training a good model. From the results shown in the last row of Table 3.14, we see that the introduced fuzzy regression transfer learning method is effective in all cases, no matter how many clusters are used. When the number of clusters is set to 5, the mean value of Q_3 in five-fold cross validation is the smallest, as is the standard deviation. As a result, using 5 clusters for this problem is the best option.

3.5 Summary

We proposed a fuzzy regression transfer learning method that modifies the input space of data through mappings to change the proportions of the hidden features for each input variables, so that the existing fuzzy rules of source domain could be more compatible for solving tasks in the target domain. Meanwhile, two different evolutionary optimization algorithms, PSO and DE, are compared and analyzed to find out the impact of the optimization procedure to the performance of the built model. Additionally, different connection ways of constructing the mappings for input space are discussed to reveal which structure of mappings is better for the domain adaptation tasks. This method effectively solves regression problems in the target domain when only a small amount of labeled data is available. Experimental results show that our method greatly improves the performance of the existing model in estimating the values of the target domain. The good performance of the proposed method on the synthetic and real-world datasets demonstrates the capability of the presented models to execute knowledge transfer with little labeled target data.

Chapter 4

Granular Domain Adaptation in Takagi-Sugeno Fuzzy Models

4.1 Introduction

Although many approaches have been introduced as possible solutions for transfer learning problems, their performance is not yet acceptable. One reason is the information granularity inherent in many problems. For instance, 128GB of mobile storage is considered large today, whereas 32GB was regarded as large five years ago. The precise values, 128GB and 32GB, both need to be expressed as a granular value, “large”, for learning to be effectively transferred from the five-year-old domain to today’s domain. Extracting additional abstract knowledge shared between domains should therefore assist knowledge transfer. Granular computing (GrC) is an emerging information processing paradigm that transforms complex data into information granules at different levels of resolution to reveal different features and irregularities. GrC’s ability to address information at different levels of abstraction could improve the performance of transfer learning, and consequently we propose several granular fuzzy regression domain adaptation methods (Zuo et al., 2017b), GFRDA for short, to address regression domain adaptation problems.

Granular computing is an emerging information processing paradigm that transforms complex data into information granules at different solution levels. Information granules can be perceived as a collection of elements drawn together by their closeness (resemblance, proximity, functionality, etc.) and articulated in terms of useful spatial, temporal, or functional relationships. Granular computing (GrC) represents, constructs, and processes information granules.

Information granules are formalized in many different ways. Depending on the problem, different formalisms to represent the information granules have been applied, such as interval sets, fuzzy sets, rough sets, and shadowed sets. The level of granularity determines the level of detail used to classify the data. Different types of knowledge can be captured or learned by representing data with information granules at different levels (Pedrycz, 2014). Features and regularities in the data can emerge, while the detail is deliberately hidden (Pedrycz, 2013). For example, interesting cloud patterns representing a cyclone may be notable in a low-resolution satellite image, while in a higher-resolution image, this large-scale atmospheric phenomenon might be missed. High resolution images are more useful for observing small-scale phenomenon, such as an interesting street pattern in Manhattan.

The most significant perspective we introduce with regard to GrC is that it is possible to obtain different levels of knowledge when dealing with data represented by information granules that have different levels of granularity. The higher the level of granularity, the more abstract the knowledge obtained.

4.2 Problem statement

The dataset in the source domain is denoted by $\boldsymbol{D} = (\boldsymbol{x}_1^s, y_1^s), (\boldsymbol{x}_2^s, y_2^s), \ldots, (\boldsymbol{x}_{N_s}^s, y_{N_s}^s)$, where $\boldsymbol{x}_k^s \in R^n, k = 1, 2, \ldots, N_s$ is the n-dimensional input variable, the label $y_k^s \in R$ is the continuous output variable, and N_s indicates the number of data. Since the amount of source data with labels is massive, a well-performing regression model for the source domain can be learned.

The dataset in the target domain contains two subsets: one with labels and one without labels $\boldsymbol{H} = \{\boldsymbol{H}_L, \boldsymbol{H}_U\} = \{\{(\boldsymbol{x}_1^t, y_1^t), \ldots, (\boldsymbol{x}_{N_{t1}}^t, y_{N_{t1}}^t)\}, \{\boldsymbol{x}_{N_{t1+1}}^t, \ldots, \boldsymbol{x}_{N_t}^t\}\}$, where $\boldsymbol{x}_k^t \in R^n, k = 1, 2, \ldots, N_t$ is the n-dimensional input variable, $y_k^t \in R, k = 1, 2, \ldots, N_{t1}$ is the continuous output variable. $\boldsymbol{H}_L$ includes the instances with labels, and $\boldsymbol{H}_U$ contains the data without labels. The numbers of data in $\boldsymbol{H}_L$ and $\boldsymbol{H}_U$ are N_{t1} and $N_t - N_{t1}$, respectively, and satisfy $N_{t1} << N_t, N_{t1} << N_s$.

The different data distributions in two domains make the usage of source model for the target tasks become impossible. Additionally, the insufficient labeled data in the the target domain cannot guarantee the accuracy of the constructed model for the target domain. Therefore, we hope to use learned knowledge of source domain to help build a fuzzy model that is compatible with target data.

4.3 The definitions of fuzzy domain adaptation

In the original definition of transfer learning, the source domain and the target domain are distinguished by the feature space, the probability distribution in the domain, and by the task, usually represented by a prediction function. In domain adaptation, which is a category of inductive transfer learning, the source domain and the target domain have the same feature space but a different distribution (Pan and Yang, 2010). Most works in the computational intelligence area on domain adaptation are based on this definition and apply neural or Bayes networks as the basic learning model.

We use a TS as the basic model for our learning tasks. Since the characteristics of this fuzzy rule-based model are not the same as a neural or Bayesian networks, domain adaptation must be redefined for fuzzy systems as follows:

Definition 4.1. (Fuzzy Domain Adaptation)

In a TS, a source domain and a target domain are represented as:

Source domain: $D_s = \{F_s = (F_1, \cdots, F_n), G_s(\boldsymbol{x}) = (G_{s1}(\boldsymbol{x}), \cdots, G_{sc}(\boldsymbol{x})), L_s = \{L_{s1}, \cdots, L_{sc}\}\}$

Target domain: $D_t = \{F_t = (F_1, \cdots, F_n), G_t(\boldsymbol{x}) = (G_{t1}(\boldsymbol{x}), \cdots, G_{tc}(\boldsymbol{x})), L_t = \{L_{t1}, \cdots, L_{tc}\}\}$

where F_s and F_t are the feature spaces in two domains. $G_{s1}, \cdots, G_{sc}$ and $G_{t1}, \cdots, G_{tc}$ are the constructed fuzzy sets in two domains, and $G_{s1}(\boldsymbol{x}), \cdots, G_{sc}(\boldsymbol{x})$ and $G_{t1}(\boldsymbol{x}), \cdots, G_{tc}(\boldsymbol{x})$ form the membership functions, which determine the condition parts of the fuzzy rules. L_s and L_t are the linear functions that govern the conclusion parts of the fuzzy rules.

In fuzzy homogeneous domain adaptation, the feature spaces in the two domains are the same, $F_s = F_t$, but either the fuzzy sets or the linear functions or both are different across the two domains, giving $G_s(\boldsymbol{x}) \neq G_t(\boldsymbol{x})$, and/or $L_s \neq L_t$. In general, we consider that $G_s(\boldsymbol{x}) = G_t(\boldsymbol{x})$ means $G_{si}(\boldsymbol{x}) = G_{ti}(\boldsymbol{x})$, $i = 1, \cdots, c$, and $G_{si}(\boldsymbol{x}) \neq G_{ti}(\boldsymbol{x})$ means $G_{si}(\boldsymbol{x}) \neq G_{ti}(\boldsymbol{x})$, $i = 1, \cdots, c$. Similarly, $L_s = L_t$ indicates $L_{si} = L_{ti}$, $i = 1, \cdots, c$, and $L_s \neq L_t$ indicates $L_{si} \neq L_{ti}$, $i = 1, \cdots, c$.

4.4 Granular domain adaptation in Takagi-Sugeno fuzzy models

A framework of granular fuzzy domain adaptation is proposed first to provide a comprehensive framework for the domain adaptation-based of fuzzy models. The methods and the corresponding algorithms are described with details to deal with different cases in fuzzy transfer learning. Finally, the performance index is given to facilitate the validation of the proposed methods.

4.4.1 Granular transfer learning framework

According to the fuzzy domain adaptation model defined above, the discrepancies between the source domain and the target domain can be summarized according to one of three cases: having different conditions, different conclusions, or both. To emphasize the difference, the source domain's model is fixed, and the target domain's model is varied according to these three cases.

Suppose fuzzy model M^s in the source domain, described in the form of the fuzzy rules, is:

model M^s

$$\text{if } \boldsymbol{x}^s \text{ is } A_i(\boldsymbol{x}^s,\ \boldsymbol{v}_i^s),\ \text{then } y \text{ is } L_i(\boldsymbol{x}^s, \boldsymbol{a}_i^s) \qquad i = 1, 2, \cdots, c \tag{4.1}$$

The target models that corrpesponds different cases are as follows:

model M_1^t

$$\text{if } \boldsymbol{x}^t \text{ is } A_i(\boldsymbol{x}^t,\ \boldsymbol{v}_i^t),\ \text{then } y \text{ is } L_i(\boldsymbol{x}^t, \boldsymbol{a}_i^s) \qquad i = 1, 2, \cdots, c \tag{4.2}$$

model M_2^t

$$\text{if } \boldsymbol{x}^t \text{ is } A_i(\boldsymbol{x}^t,\ \boldsymbol{v}_i^s),\ \text{then } y \text{ is } L_i(\boldsymbol{x}^t, \boldsymbol{a}_i^t) \qquad i = 1, 2, \cdots, c \tag{4.3}$$

model M_3^t

$$\text{if } \boldsymbol{x}^t \text{ is } A_i(\boldsymbol{x}^t,\ \boldsymbol{v}_i^t),\ \text{then } y \text{ is } L_i(\boldsymbol{x}^t, \boldsymbol{a}_i^t) \qquad i = 1, 2, \cdots, c \tag{4.4}$$

In the first case, comparing models M^s and M_1^t, the conditions of the fuzzy rules in the two domains are different, but the conclusions are the same. In the second case, comparing models M^s and M_2^t, the conditions of the fuzzy rules are the same, but the conclusions are totally different. In the third case, comparing models M^s and M_3^t, the conditions and conclusions of the fuzzy rules in both the source and target domains are different.

There are now massive amounts of labeled data in the source domain and a well-performing model M^s can be built. In the target domain, there is a large amount of unlabeled data and little labeled data, so establishing a prediction model is impossible. Because the fuzzy rules in the two domains are different, the model for the source domain M^s is not suited to regression tasks in the target domain.

Next, domain adaptation problems are analyzed from the perspective of GrC. The knowledge contained in both the source and target domains can be treated as information granules. Since the information granules in each domain have different levels of granularity, a model based solely on knowledge from the source domain could not directly solve tasks in the target domain. For example, RAM is an important index for predicting the price of a computer. Thirty years ago, computers typically had 256kb of RAM, whereas now 8G is fairly standard. These two values, 256kb and 8G, can both be treated as information granules, but with different granularity levels as their unit of measurement are different. Therefore, the knowledge gleaned from data based on 256kb is not suitable for tasks relevant to the 8G information.

The higher the granularity level in GrC, the more abstract the knowledge extracted. Based on the existing knowledge (information granules) in the source domain, our idea is to extract and construct information granules at a higher level of granularity so that knowledge can be appropriately shared between the two domains. However, the knowledge contained in the new information granules cannot be directly used to solve tasks in the target domain since the required level of granularity is different. An additional procedure is needed to transform the new information granules to a lower level, so they can be applied to help solve the target tasks. The essence of this process is shown in Fig. 4.1.

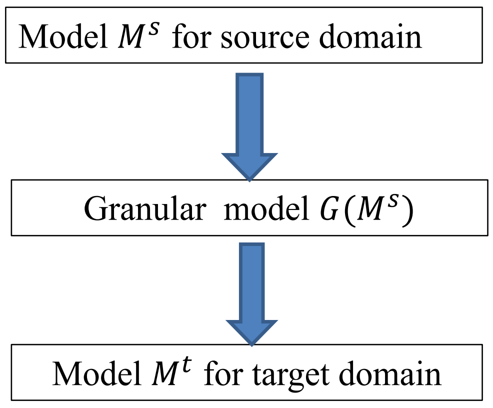


Figure 4.1 Knowledge transfer from a GrC perspective

The process has two steps. A granular model is built by transforming the information granules from a lower level to a higher level, and the granularity level of the new granules is reduced to suit the target domain.

Continuing the example of the computer's internal storage, our aim is to use a more abstract representation to describe a computer's RAM 30 years ago. For instance "large capacity" instead of a numerical value: 256kb. We can still say a computer of today has "large capacity" if its RAM meets or exceeds 8G. In this example, "large capacity" is treated as an information granule with a higher granularity level that builds a bridge to connect two granules of lower level.

Instead of conducting the two steps in Fig. 4.1 separately, we implement them simultaneously. Because the results of the first step significantly impact the performance of the following procedure, merging the two steps benefits the method's execution. A nonlinear space transformation is used to achieve these two steps, and an optimization process makes the resulting model more compatible with the tasks in the target domain.

Different strategies are applied in the three domain adaptation cases to implement the above process. Where the conditions of the fuzzy rules in two domains are different, the conditions are changed using a space transformation so that the transformed fuzzy rules approximate the expected model in the target domain. Where the conclusions of the fuzzy rules are different, the conclusions are changed using mapping to ensure the newly constructed model is as close as possible to the expected target model. Where both the conditions and the conclusions are different, a method that modifies both the conditions and conclusions is used so that the transformed fuzzy rules approximate the expected fuzzy rules and are more compatible with the target domain.

Since the conditions of fuzzy rules are governed by the input data, the method that changes the conditions can be regarded as transforming the input space. The conclusions determine the output of the fuzzy rules, so this method transforms the output space. Similarly, the last method transforms both the input and output spaces.

A simple example with one-dimensional input data is shown in Fig. 4.2 to illustrate space transformation in our proposed model.

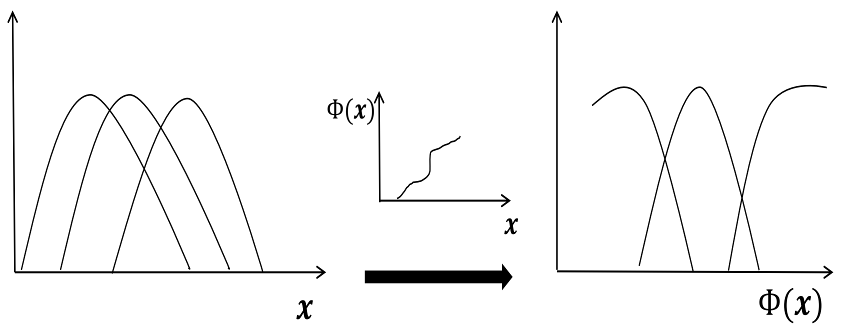


Figure 4.2 An example of conditions of fuzzy rules under space transformation

The left section of Fig. 4.2 shows the membership functions of the fuzzy sets constructed with FCM. Using the space transformation Φ, the input variable $\boldsymbol{x}$ becomes $\Phi(\boldsymbol{x})$. More importantly, the membership functions in the new space have been changed, as shown on the right.

This section describes the specific procedures of the proposed Granular Fuzzy Regression Domain Adaptation (GFRDA) methods, followed by the performance index used to evaluate the constructed models.

4.4.2 Methods and algorithms of GFRDA

In the proposed GFRDA methods, the process of transferring knowledge from the source domain to the target domain has two steps. First a TS fuzzy model based on source data is constructed; second, a new fuzzy model for the target domain is built by modifying the input and/or output space of the existing model (fuzzy rules). The first step is the same for all three methods, while the second step differs depending on the method. This process is shown in Fig. 4.3.

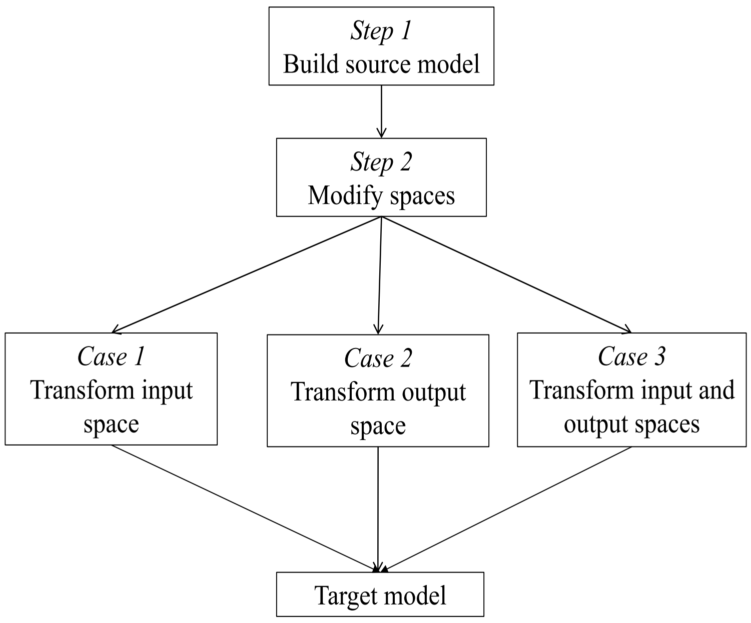


Figure 4.3 The granular fuzzy domain adaptation process

The procedure for Step 1, building a fuzzy model for the source domain, follows.

Step 1: Construct a TS fuzzy model M^s based on source data.

A TS fuzzy model M^s is constructed using source data $\boldsymbol{D}$.

model M^s

$$\text{if } \boldsymbol{x}_k^s \text{ is } A_i(\boldsymbol{x}_k^s,\ \boldsymbol{v}_i^s),\text{ then } y_k^s \text{ is } L_i(\boldsymbol{x}_k^s, \boldsymbol{a}_i^s) \qquad i = 1, 2, \cdots, c \tag{4.5}$$

The main blocks of fuzzy rules are the conditions and conclusions, which are dominated by prototypes of the data and linear functions, respectively. Model M^s is therefore constructed by calculating the data prototypes and the linear functions. Thus, we have the prototypes $\boldsymbol{v}_1^s, \cdots, \boldsymbol{v}_c^s$, and the linear functions $L_1(\cdot, \boldsymbol{a}_1^s), \cdots, L_c(\cdot, \boldsymbol{a}_c^s)$.

We now take some data from the dataset $\boldsymbol{H}$ in the target domain; however, model M^s does not perform well on dataset $\boldsymbol{H}$, since these data follow a different fuzzy model and different fuzzy rules to those of model M^s. The number of labelled data in dataset $\boldsymbol{H}_L$ is not sufficiently large to build a good model for the target domain, so the proposed methods apply knowledge from the source domain to help the target domain build a new model.

In the second step, the input and/or output space of model M^s obtained in Step 1 is modified through mappings using the labeled target data $\boldsymbol{H}_L$ to build a new fuzzy model for the target domain.

Step 2: Modify the existing fuzzy rules to build a new fuzzy regression model for the target domain.

The three different homogeneous domain adaptation cases, shown in 4.2 - 4.4, are considered, and the steps for the corresponding GFRDA method are explained below.

Step 2a) Method 1: change the input space

To handle the cases where the fuzzy rules' conditions in the source and target domains are not identical, we apply the method proposed in Chapter 3 Zuo et al. (2017a). The target domain's ideal model is described in 4.2. Since the way of constructing the space transformation is the same in these three methods, we detail it in this method and not repeat in the other two methods.

Since there is insufficient labeled data to train the fuzzy model M_1^t, the learned knowledge (fuzzy rules) in the existing model M^s is used to help the target domain construct a new fuzzy model. In this method, the input space is changed by optimizing a continuous mapping for each input variable. Through mapping Φ, the input space is transformed to $\Phi(\boldsymbol{x}^t)$, and the new fuzzy model $\overline{M}_1^t$ for the target domain is constructed using the fuzzy rules from model M^s. This process and resulting architecture are shown in Fig. 4.4.

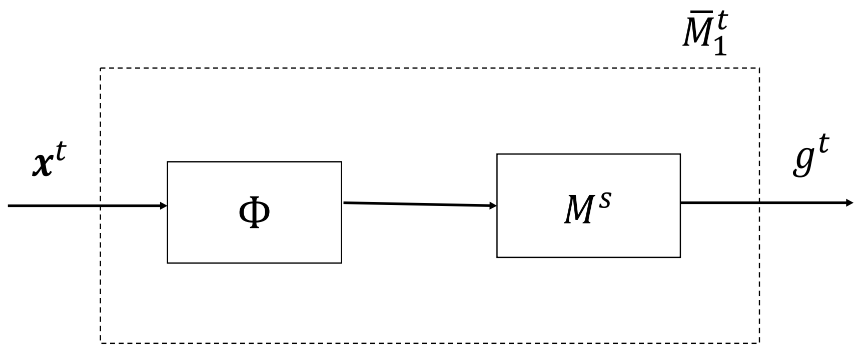


Figure 4.4 Method 1: changing the input space

model $\overline{M}_1^t$

$$\text{if } \boldsymbol{x}_k^t \text{ is } A_i(\Phi(\boldsymbol{x}_k^t),\ \boldsymbol{v}_i^s), \text{ then } y_k \text{ is } L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s) \qquad i = 1, 2, \cdots, c \tag{4.6}$$

Because mapping Φ is a transformation of the input space, the changes are reflected in the input data and the prototypes (the centers of the clusters) with the forms $\Phi(\boldsymbol{x}_k^t)$ and $\Phi(\boldsymbol{v}_i^s)$.

Therefore the output of model $\overline{M}_1^t$ is:

$$g_k^t = \sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}_k^t),\ \boldsymbol{v}_i^s) L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s) \tag{4.7}$$

Our aim is to find a Φ such that $\overline{M}_1^t$ becomes compatible with the target data, i.e.,

$$\sum_{k=1}^{N_t} \sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}_k^t),\ \boldsymbol{v}_i^s) L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s) \approx \sum_{k=1}^{N_t} y_k^t \tag{4.8}$$

The parameters of Φ are optimized by minimizing the objective function as follows:

$$Q_1^t = \sqrt{\frac{1}{N_{t1}}\sum_{k=1}^{N_{t1}}\sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}_k^t),\ \boldsymbol{v}_i^s)L_i(\Phi(\boldsymbol{x}_k^t),\boldsymbol{a}_i^s) - y_k^t)^2} + \frac{\lambda}{2}w^T w \tag{4.9}$$

The first term in 4.9 is the approximation error that aims to minimize the gap between the output of model $\overline{M}_1^t$ and the target data's real output. The second term introduces a structural risk term into the objective function. The parameter λ indicates the tradeoff between the quality of an approximation and the complexity of the approximation function; w is the vector of all the parameters optimized.

The mapping is the key element in each of our GFRDA methods. We use nonlinear continuous functions, composed of sigmoid functions, to construct the mappings of Φ.

Similarly, the nonlinear mappings based on sigmoid functions are used to change the input space of the target domain. The mapping is composed of P nodes in the hidden layer and a single node at the output layer that constructs the network. The two parametric sigmoid functions are applied to the hidden nodes, where the pth sigmoid function for the jth input variable of $\boldsymbol{x}_k^t$ is:

$$z_{kjp} = \frac{1}{1+e^{-\alpha_{jp}(x_{kj}^t - \beta_{jp})}} \tag{4.10}$$

where $j = 1,\ldots,n$, $p = 1,\ldots,P$, $\alpha_{jp} > 0$.

Therefore, the transformation of input variable x_{kj}^t under mapping Φ_j is:

$$\Phi_j(x_{kj}^t) = \sum_{p=1}^{P} w_{jp} * z_{kjp} \tag{4.11}$$

where w_{jp} represents the weight of the pth sigmoid function of the output variable, and satisfies $\sum\limits_{p=1}^{P} w_{jp} = \max\{x_{1j}^t,\cdots,x_{N_tj}^t\}$. z_{kjp} is calculated through 4.10.

$\Phi = [\Phi_1 \Phi_2 \cdots \Phi_n]$, where Φ_j is constructed following the procedure described above. Thus the input data $\boldsymbol{x}_k^t$ becomes $\Phi(\boldsymbol{x}_k^t)$:

$$\Phi\left(\boldsymbol{x}_k^t\right) = \begin{bmatrix} \Phi_1\left(x_{k1}^t\right) \\ \Phi_2\left(x_{k2}^t\right) \\ \cdots \\ \Phi_n\left(x_{kn}^t\right) \end{bmatrix} = \begin{bmatrix} \sum_{p=1}^{P} w_{1p} \frac{1}{1+e^{-\alpha_{1p}\left(x_{k1}^t - \beta_{1p}\right)}} \\ \sum_{p=1}^{P} w_{2p} \frac{1}{1+e^{-\alpha_{2p}\left(x_{k2}^t - \beta_{2p}\right)}} \\ \cdots \\ \sum_{p=1}^{P} w_{np} \frac{1}{1+e^{-\alpha_{np}\left(x_{kn}^t - \beta_{np}\right)}} \end{bmatrix} \tag{4.12}$$

Taking advantage of the nonlinear mappings, transformations are made to the input space so that the new input variables become more compatible with the data in the target domain. The parameters of Φ are derived through an optimization process by minimizing 4.9 using the labeled dataset $\boldsymbol{H}_L$. The dataset $\boldsymbol{H}_U$ is used to test the performance of the model after its construction.

Step 2b) Method 2: change the output space

This method handles cases where the conclusions of the fuzzy rules in the two domains are different. The target domain's ideal model is described in 4.3.

Again, there are insufficient data to train the fuzzy model M_2^t for the target domain. Since the conclusions are different to those in the source domain, we introduce a method to modify the output space by optimizing a continuous mapping for each output. The output space is modified by mapping Ψ, and a new fuzzy model $\overline{M}_2^t$ for the target domain is constructed based on the fuzzy rules in model M^s. The process and resulting architecture are shown in Fig. 4.5.

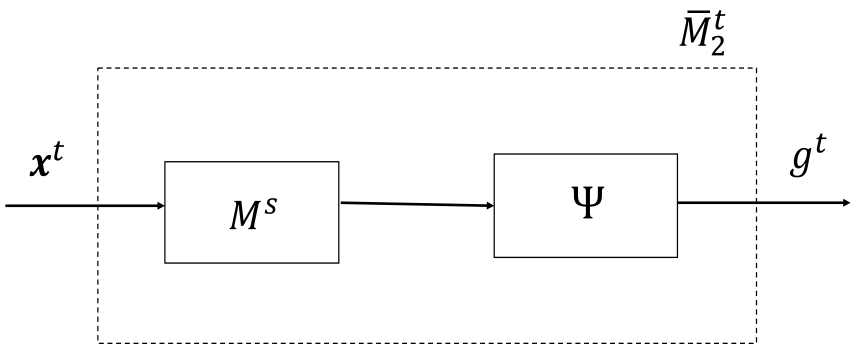


Figure 4.5 Method 2: changing the output space

model $\overline{M}_2^t$

$$\text{if } \boldsymbol{x}_k^t \text{ is } A_i(\boldsymbol{x}_k^t,\ \boldsymbol{v}_i^s), \text{ then } y_k^t \text{ is } \Psi_i(L_i(\boldsymbol{x}_k^t, \boldsymbol{a}_i^s)) \qquad i = 1, 2, \cdots, c \tag{4.13}$$

Therefore, the output of model $\overline{M}_2^t$ is:

$$g_k^t = \sum_{i=1}^{c} A_i(\boldsymbol{x}_k^t,\ \boldsymbol{v}_i^s)\Psi_i(L_i(\boldsymbol{x}_k^t, \boldsymbol{a}_i^s)) \tag{4.14}$$

Our aim is to find a Ψ such that $\overline{M}_2^t$ becomes compatible with the target data, i.e.,

$$\sum_{k=1}^{N_t}\sum_{i=1}^{c} A_i(\boldsymbol{x}_k^t,\ \boldsymbol{v}_i^s)\Psi_i(L_i(\boldsymbol{x}_k^t, \boldsymbol{a}_i^s)) \approx \sum_{k=1}^{N_t} y_k^t \tag{4.15}$$

The parameters of Ψ are optimized by minimizing the objective function as follows:

$$Q_2^t = \sqrt{\frac{1}{N_{t1}}\sum_{k=1}^{N_{t1}}\sum_{i=1}^{c} A_i(\boldsymbol{x}_k^t,\ \boldsymbol{v}_i^s)\Psi_i(L_i(\boldsymbol{x}_k^t, \boldsymbol{a}_i^s)) - y_k^t)^2} + \frac{\lambda}{2}w^T w \tag{4.16}$$

The construction of mapping Ψ is similar to the construction of mapping Ψ in Method 1. $\Psi = [\Psi_1 \Psi_2 \cdots \Psi_c]$, and the parameters are obtained by minimizing 4.16 using the labeled dataset $\boldsymbol{H}_L$; w represents the vector of all the parameters optimized.

Step 2c) Method 3: changing both the input and output spaces

In this case, both the conditions and the conclusions of the fuzzy rules in the two domains are different. The target domain's ideal model is described in 4.4.

This method is a combination of the first two and uses the mappings to modify the input and output spaces. The input space is transformed to $\Phi(\boldsymbol{x}^t)$ by mapping Φ, the output space is transformed by mapping Ψ, and the new fuzzy model $\overline{M}_3^t$ for the target domain is constructed based on the fuzzy rules in model M^s. The process and resulting architecture are shown in Fig. 4.6.

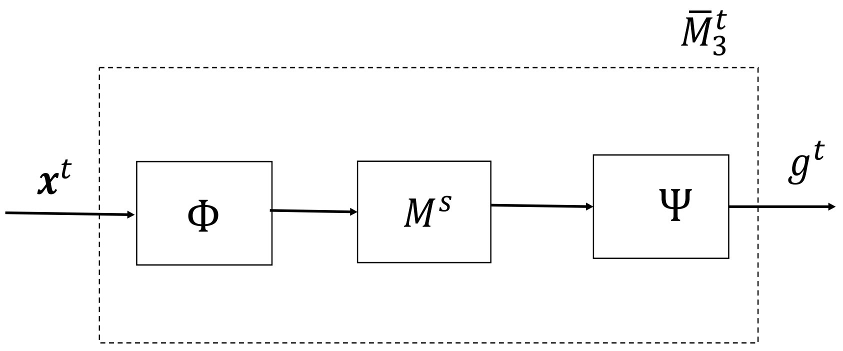


Figure 4.6 Method 3: changing both the input and output spaces

Model $\overline{M}_3^t$, described in the form of fuzzy rules, is:

model $\overline{M}_3^t$

$$\text{if } \boldsymbol{x}_k^t \text{ is } A_i(\Phi(\boldsymbol{x}_k^t),\ \boldsymbol{v}_i^s),\ \text{then } y_k \text{ is } \Psi_i(L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s)) \qquad i = 1, 2, \cdots, c \tag{4.17}$$

Therefore, the output of model $\overline{M}_3^t$ is:

$$g_k^t = \sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}_k^t),\ \boldsymbol{v}_i^s)\Psi_i(L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s)) \tag{4.18}$$

Our aim is to find a Φ and Ψ such that $\overline{M}_3^t$ becomes compatible with the target data, i.e.,

$$\sum_{k=1}^{N_t} \sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}_k^t),\ \boldsymbol{v}_i^s)\Psi_i(L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s)) \approx \sum_{k=1}^{N_t} y_k^t \tag{4.19}$$

The parameters of Φ and Ψ are optimized by minimizing the objective function as follows:

$$Q_3^t = \sqrt{\frac{1}{N_{t1}} \sum_{k=1}^{N_{t1}} \sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}^t),\ \boldsymbol{v}_i^s)\Psi_i(L_i(\Phi(\boldsymbol{x}^t), \boldsymbol{a}_i^s)) - y_k^t)^2} + \frac{\lambda}{2} w^T w \tag{4.20}$$

The construction of mappings Φ and Ψ is exactly the same as that in Methods 1 and 2, and the parameters of Φ and Ψ are optimized by minimizing 4.20 using the labeled dataset $\boldsymbol{H}_L$. Similarly, the objective function includes two terms: the approximation error and the structural risk.

4.4.3 Performance index

Another model is also trained using insufficient data in the target domain. Although there is only a small amount of labeled data in the target domain, they can still be used to train a model. Proving that a model does not perform as well when trained with less data in the target domain supports our assumption. As a result, three models are constructed: the first is built using the source data for the source domain (model M^s); the second is built using the insufficient target data for the target domain (model $\tilde{M}^t$); and the third is built using the proposed granular fuzzy domain adaptation methods $\overline{M}^t$ (models $\overline{M}_1^t, \overline{M}_1^t, \overline{M}_1^t$, corresponding to three cases).

The datasets in the source domain and the target domain are $\boldsymbol{D}$ and $\boldsymbol{H}$, as described in the above subsection. When constructing the above models, we used a five-fold cross validation procedure, which is commonly used to validate models in machine learning. Dataset $\boldsymbol{D}$ is split into a training set $\boldsymbol{D}_1$ (80%) and a testing set $\boldsymbol{D}_2$ (20%). The number of data in $\boldsymbol{D}$, $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are N_s, N_{s1}, and N_{s2}, respectively. Similarly, the labeled data in $\boldsymbol{H}_L$ are split into a training set $\boldsymbol{H}_{L1}$ (80%) with N_{t11} data and a testing set $\boldsymbol{H}_{L2}$ (20%) with N_{t12} data.

Symbolic representations of the models' performance follow.

Model's performance in the source domain is represented by Q, which is the **r**oot **m**ean **s**quare **e**rror (RMSE) of the expected value and the output from model M^s.

$$Q = \sqrt{\frac{1}{N_{s2}} \sum_{k=1}^{N_{s2}} (d_k^s - y_k^s)^2} \tag{4.21}$$

where d_k^s is the output of model M^s when its input is $\boldsymbol{x}_k^s$, $\boldsymbol{x}_k^s \in \boldsymbol{D}_2$.

For consistency, dataset $\boldsymbol{H}_U$ is used to test the model's performance in the target domain in the following discussion.

The performance of model M^s in the target domain is denoted by Q_1, which indicates the ability of the source domain's model to address tasks in the target domain.

$$Q_1 = \sqrt{\frac{1}{N_t - N_{t1}} \sum_{k=N_{t1}+1}^{N_t} (h_k^t - y_k^t)^2} \tag{4.22}$$

where h_k^t is the output of model M^s when the input is $\boldsymbol{x}_k^t$, $\boldsymbol{x}_k^t \in \boldsymbol{H}_U$, and y_k^t is the expected output to input $\boldsymbol{x}_k^t$.

The insufficient labeled data in the target domain $\boldsymbol{H}_L$ are used to train a model $\tilde{M}^t$ for the target domain using the same construction procedures as model M^s for the source domain. The performance of model $\tilde{M}^t$ in the target domain is denoted as Q_2:

$$Q_2 = \sqrt{\frac{1}{N_t - N_{t1}} \sum_{k=N_{t1}+1}^{N_t} (s_k^t - y_k^t)^2} \tag{4.23}$$

where s_k^t is the output of model $\tilde{M}^t$ when its input is $\boldsymbol{x}_k^t$, $\boldsymbol{x}_k^t \in \boldsymbol{H}_U$, and y_k^t is the expected output to input $\boldsymbol{x}_k^t$.

The model $\overline{M}_1^t, \overline{M}_2^t, \overline{M}_3^t$, constructed using our GFRDA methods, is also tested on the target dataset $\boldsymbol{H}_U$, and the result is denoted as Q_3:

$$Q_3 = \sqrt{\frac{1}{N_t - N_{t1}} \sum_{k=N_{t1}+1}^{N_t} (r_k^t - y_k^t)^2} \tag{4.24}$$

where r_k^t is the output of model $\overline{M}^t$ when the input is $\boldsymbol{x}_k^t$, $\boldsymbol{x}_k^t \in \boldsymbol{H}_U$, y_k^t is the expected output to input $\boldsymbol{x}_k^t$.

When constructing the model $\overline{M}^t$ for the target domain, a DE algorithm is used to optimize the parameters of the mappings and build the new GFRDA models for the target domain. DE is a computational method that determines an optimal solution by iteratively navigating a population of solutions, which minimizes a certain predetermined objective function. Such methods are commonly known as metaheuristics, as they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions Price et al. (2006); Storn and Price (1997). PSO is another famous evolutionary algorithm. Based on the experimental results in Chapter 3 Zuo et al. (2017a), the algorithmic stability of DE is superior to PSO, so DE was selected as the optimization algorithm for the models' construction. In DE, there are two parameters that largely influence optimization performance: the differential weight F and the crossover probability CR. The value range of F is $[0,2]$, and the value range of CR is $[0,1]$. In addition, due to the problem's complexity, the same initialization strategy is used in all the experiments below: 200 candidate solutions are generated, and the maximum number of iterations is set to 200.

The values of Q_1, Q_2 and Q_3 are compared in the following experiments. The desired outcome is that Q_3 should be smaller than both Q_1 and Q_2. $Q_3 < Q_1$ demonstrates that the performance of the new constructed model $\overline{M}^t$ on the target domain is superior to the existing model M^s, and $Q_3 < Q_2$ shows that the model $\tilde{M}^t$ trained using a few labeled data has poor performance compared to model $\overline{M}^t$.

4.5 Empirical results analysis

Both synthetic and real-world datasets were used to evaluate the proposed GFRDA methods and their algorithms. Except for the domain adaptation problems for regression tasks, the proposed methods are also used to solve a special scenario in classification problem, the label space adaptation.

4.5.1 Experiments on synthetic datasets

This section consists of three experiments to discuss and analyze the effectiveness of the proposed GFRDA methods. The first experiment validates the presented methods and analyzes the impact of an important parameter in the performance of the constructed models – the trade-off parameter λ. The second experiment explores the effect on the results when the number of the labeled target data changes. The third experiment compares the outcomes of the three methods when dealing with different cases in domain adaptation problems.

4.5.1.1 Datasets and experimental settings

The datasets used in these three experiments contain some repetition, so all the datasets are described first, followed by the details of their application.

The datasets are generated by the input data and the linear functions. The conditions of the fuzzy rules are governed by the centers of the clusters, which decide the membership functions of the constructed fuzzy sets. Since FCM is used to build the clusters and the fuzzy sets, the cluster centers are significantly affected by the distribution of the input data. Therefore, to obtain the source and target data with different cluster centers, the input data in the source and target domains should be generated with different distributions.

Two groups of input data with different distributions are shown in Table 4.1, and similarly two groups of linear functions with disparate parameters are displayed in Table 4.1.

Table 4.1 contains two groups of input data, input data 1 and input data 2; the method used to generate the data was the same for both groups. To obtain more than one fuzzy rule and differentiate between the cluster centers, we generated three sub-datasets by varying the distribution and combining them to construct the whole dataset. The first sub-dataset was generated using normalized distribution $N(\boldsymbol{\mu}_1, \boldsymbol{\sigma}_1)$, and the other two sub-datasets were built in the same way.

Table 4.2, lists the two groups of linear functions with different parameters. There are three linear functions in each group that correspond to the input data in Table 4.1.

Table 4.1 The distributions of input data 1 and data 2

Input data 1		Input data 2	
Mean values	Covariance	Mean values	Covariances
$\mu_1 = [1\ 1]$	$\sigma_1 = \begin{pmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{pmatrix}$	$\mu_4 = [2.5\ 1.5]$	$\sigma_4 = \begin{pmatrix} 0.4^2 & 0.1 \\ 0.1 & 0.4^2 \end{pmatrix}$
$\mu_2 = [2\ 1]$	$\sigma_2 = \begin{pmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{pmatrix}$	$\mu_5 = [2\ 1.5]$	$\sigma_5 = \begin{pmatrix} 0.5^2 & 0.1 \\ 0.1 & 0.5^2 \end{pmatrix}$
$\mu_3 = [1.5\ 2]$	$\sigma_3 = \begin{pmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{pmatrix}$	$\mu_6 = [2.5\ 2]$	$\sigma_6 = \begin{pmatrix} 0.5^2 & 0.1 \\ 0.1 & 0.5^2 \end{pmatrix}$

Table 4.2 Coefficients of linear functions in the two different groups

Input data 1		input data 2	
$L_1(\boldsymbol{a}_1)$	$\boldsymbol{a}_1 = [1\ 1\ 1]$	$L_4(\boldsymbol{a}_4)$	$\boldsymbol{a}_4 = [2\ 0.5\ 1.5]$
$L_2(\boldsymbol{a}_2)$	$\boldsymbol{a}_2 = [2\ 2\ 1]$	$L_5(\boldsymbol{a}_5)$	$\boldsymbol{a}_5 = [1\ 2\ 0.5]$
$L_3(\boldsymbol{a}_3)$	$\boldsymbol{a}_3 = [-1\ 1\ 3]$	$L_5(\boldsymbol{a}_6)$	$\boldsymbol{a}_6 = [-1.5\ 2\ 4.5]$

Various combinations of the input data in Table 4.1 and the linear functions in Table 4.2 result in different datasets. Thus, three datasets were constructed, as shown in Table 4.3, to represent the three different cases in domain adaptation described in Section 4.4.

Table 4.3 Datasets for three cases in domain adaptation

	Source domain	Target domain
Dataset 1	Input data 1 + Linear function 1	Input data 2 + Linear function 1
Dataset 2	Input data 1 + Linear function 1	Input data 1 + Linear function 2
Dataset 3	Input data 1 + Linear function 1	Input data 2 + Linear function 2

From Table 4.3, we can see that the dataset in the source domain is fixed, and the varying dataset in the target domains lead to three different cases. Constructing the target data using input data 2 and linear function 1 reflects cases where the input data differs between the two domains. Using input data 1 and linear function 2, reflects cases where the conclusion of the fuzzy rules are not the same, and using input data 2 and linear function 2 reflects differences in both the conditions and conclusions.

4.5.1.2 Experimental results

Experiment A: Verifying the proposed GFRDA methods.

The purpose of this subsection is to verify the ability of the proposed methods to solve three cases in domain adaptation, and further explore the impact of the parameter λ on the performance of the models.

Three experiments were conducted to test the methods' performance in the different domain adaption cases using the three datasets in Table 4.3. Additionally, comparing the models' performance with varying parameter λ was used to determine the optimal λ. There are 1500 labeled data in the source domain, and 15 labeled and 585 unlabeled data in the target domain.

The results and analysis of these three experiments are discussed in detail below.

a) Method 1: change the input space

This experiment changed the input space using Dataset 1 from Table 4.3 to deal with domain adaptation cases where the fuzzy rule conditions differ. Moreover, comparing model performance with different values for parameter λ was used to determine the optimal λ. The results are shown in Table 4.4.

Table 4.4 Results of the first method by varying λ

λ	Q	Q_1	Q_2	Q_3
0	0.08±0.01	1.88 ± 0.01	7730.09 ± 163972.14	1.13 ± 0.22
0.1				1.06 ± 0.07
0.2				1.04 ± 0.06
0.3				1.04 ± 0.05
0.4				1.04 ± 0.04
0.5				1.05 ± 0.04
0.6				1.06 ± 0.03
1				1.15 ± 0.04
2				3.41 ± 0.38

Because five-fold cross validation was used, all the values for Q, Q_1, Q_2 and Q_3 are written in the form of "mean $\pm$ standard deviation". Since changing λ only impacts the construction of model $\overline{M}_1^t$, the values of Q, Q_1 and Q_2 that are related to model M^s and $\tilde{M}^t$ are constant under different λ values. From Table 4.4 we can see that the mean value of Q_1 is 1.88, which indicates that the model of the source domain does not fit the target data very well. The number of labeled target data is small, resulting in a very large mean value of 77730.09 and a large standard deviation of 163972.14 for Q_2 , to represent model $\tilde{M}^t$'s performance. However, when λ is not bigger than 1, the mean values for Q_3 are smaller than those of Q_1 and Q_2. This indicates that the model built using our method is superior to the source domain's model and the model constructed using the target data. When λ is greater than 0.4, the values of Q_3 appear to have a growth trend and is lowest when λ is equal to 0.4.

b) Method 2: change the output space

Dataset 2 was used to simulate cases where the conclusions of the fuzzy rules differ by changing the output space. The results are shown in Table 4.5.

Table 4.5 Results of the second method by varying λ

λ	Q	Q_1	Q_2	Q_3
0	0.50 ± 0.02	3.37 ± 0.00	62496.40 ± 122924.89	1.84 ± 0.28
0.01				1.85 ± 0.24
0.02				1.89 ± 0.26
0.05				2.06 ± 0.30
0.1				3.06 ± 0.29
1				8.90 ± 0.08

Compared to the last experiment, the values of Q_3 are sensitive to changes in λ, and tends to increase with an increase in λ. When λ is smaller than 0.1, the mean values of Q_3 are no greater than the mean values of Q_1 and Q_2, which shows the superiority of our proposed method. Model $\overline{M}_1^t$ shows the best performance when λ is set to 0.

c) Method 3: change the input and output spaces

Dataset 3 was used to test cases where both the conditions and conclusions of the fuzzy rules differ; therefore, both the input and output spaces were transformed. The results are shown in Table 4.6.

Table 4.6 Results of the third method by varying λ

λ	Q	Q_1	Q_2	Q_3
0	0.08± 0.01	3.28± 0.02	2496.40± 122924.89	2.32± 0.37
0.01				2.85± 0.10
0.02				2.92± 0.29
0.05				3.448± 0.30
0.1				4.05± 0.19
1				9.00± 0.07

From Table 4.6, we can see that similar to the last experiment, a tiny change in λ results in and increase in Q_3. When λ is not smaller than 0.05, the mean value of Q_3 is greater than that of Q_1, which means the model using the proposed method is not better than the source domain model. However, the proposed method works well when the value of λ is small.

Experiment B: Exploring the impact of the number of labeled target data.

In the above experiments, the number of labeled target data was fixed at 15. Since the optimization of the models $\overline{M}_1^t, \overline{M}_2^t$, and$\overline{M}_3^t$ is totally based on labeled target data, they play an important role in the ability of the constructed model to fulfill the target tasks.

This experiment was designed to analyze the performance of the constructed model with different numbers of labeled target data. The total number of data in the target domain is 1500, but the number of labeled data varies.

Here, we only transform the input space, as an example and list the results in Table 4.7.

Table 4.7 The values of Q, Q_1, Q_2, Q_3 with different number of labeled target data

N^t	Q	Q_1	Q_2	Q_3
10	0.08 ± 0.01	1.88 ± 0.01	507710.47 ± 940131.29	1.07 ± 0.07
15		1.88 ± 0.01	77730.09 ± 163972.14	1.04 ± 0.05
20		1.89 ± 0.01	17.23 ± 33.47	1.04 ± 0.04
25		1.88 ± 0.01	0.85 ± 0.54	1.01 ± 0.03
30		1.89 ± 0.01	0.76 ± 0.12	1.02 ± 0.03

As Q represents the performance of the source model on the source data, changing the number of labeled target data N_{t1} has no impact; the value of Q is constant at different N_{t1}. The values of Q_1, which represents the performance of source model on unlabeled target data with number $N_t - N_{t1}$, have tiny fluctuations, which indicates that changes in N_{t1} only slightly influence Q_1. The mean value and standard deviation of Q_2 decrease with a greater amount of labeled target data. This is because more training data is available in the target data, and model $\overline{M}_1^t$ is able to achieve better generalization of the unlabeled target data. Even though few labeled target data are available, the values for Q_3 are smaller than that of Q_1 and Q_2, which indicates that our proposed method works well in this domain adaptation problem. When the number of labeled target data is beyond 25, the proposed method does not show superiority. However, given our central assumption that the labeled target data are insufficient to construct a good model, the results obtained are reasonable.

Experiment C: Comparing the performance of the proposed methods.

Experiments on A and B show that each of the proposed methods is effective solutions to their respective domain adaptation problems. However, we were also curious about each method's ability to solve the other two cases and designed an experiment to compare the performance of all three methods in all three cases.

We must highlight that the purpose of this experiment is not to determine the best method for each case. First, the performance of these methods depends heavily on the datasets, so results from one dataset do not prove the validity of these methods in the given case. Second, we have already proven that each case is well-handled by its specifically designed method. However, if either of the other two methods also has a good performance on knowledge transfer, they can be treated as 'assistant' methods.

The three methods are used to solve the three cases in domain adaptation problems, and the results are displayed in Table 4.8.

Table 4.8 Comparison of the three methods in three cases

	Case 1	Case 2	Case 3
Q	0.08 ± 0.01	0.50 ± 0.02	0.08 ± 0.01
Q_1	1.88 ± 0.01	3.37 ± 0.00	3.28 ± 0.02
Q_2	77730.09 ± 163972.14	62496.40 ± 122924.89	62496.40 ± 122924.89
Q_3 (method 1)	1.04 ± 0.05	4.00 ± 0.17	3.07 ± 0.14
Q_3 (method 2)	1.01 ± 0.18	1.84 ± 0.28	1.85 ± 0.13
Q_3 (method 3)	1.22 ± 0.32	1.88 ± 0.16	2.32 ± 0.37

From the results shown in Table 4.8, we can see that all three methods perform well in the first case. In the second case, changing the output space or changing both the input and output spaces can also solve this domain adaptation problem. Similarly, all three methods are also valid for the third case, just not as well as the other two.

Based on the results, we conclude that the specific method designed for each case shows superior performance; furthermore, the other methods can be used as alternatives.

4.5.2 Experiments on real-world datasets

Three real-world datasets from the UCI Machine Learning Repository were used to validate the effectiveness of the proposed GFRDA methods. However, information about which case each datasets reflects is not readily available, so we use the three methods to solve this problem and discover which method was the most effective.

The "concrete compressive strength (CCS)" dataset contains eight input features to predict the concrete compressive strength output feature. The dataset was revised in two ways to make it appropriate for use in a transfer learning problem. First, the dataset was split into a source domain and a target domain based on the input feature "age": instances with an age smaller than 100 fell into the source domain, and the remaining instances were treated as data in the target domain. Second, the input features "blast furnace slag", "fly ash", and "superplasticizer" were perturbed with random numbers using the normal distributions $N(0.1, 0.1)$ in the source data and $N(5, 1)$ for the target data. There are 900 labeled instances in the source domain, and 30 labeled and 80 unlabeled instances in the target domain.

The "housing dataset (HD)" aims to predict the "MEDV" using six input attributes. The data was normalized and split into two datasets using the attribute "TAX", which represents the full-value property-tax rate per \$10,000. Instances of "TAX" smaller than 0.5 were used to form the source dataset, and instances of "TAX" larger than 0.5 were used as the target dataset. The attributes "RM", "AGE", and "B" of the source data were perturbed by random numbers coming from $N(0.1, 0.1)$, while those attributes in the target data were perturbed by normal random numbers using the distributions $N(7, 1)$, $N(5, 1)$ and $N(8, 1)$, respectively. There are 360 labeled instances in the source domain and 130 instances in the target data with 15 labeled.

The “Istanbul stock exchange (ISE)” dataset aims to predict the “MSCI emerging markets index” using the attributes “stock exchange returns” and “Istanbul stock exchange national 100 index”. The data was normalized and split into two datasets. The first 300 instances were used to form the source domain, and the next 120 instances were chosen as the target domain. Further, the two attributes were perturbed with random numbers using the normal distributions $N(0.1, 0.1)$ in the source data and $N(5, 1)$ for the target data.

The last dataset concerns “air quality (AQ)”. From the provided attributes, we selected two attributes, “temperature” and “relative humidity”, as the input data, and chose “absolute humidity” as the output. All the attributes were normalized, and the dataset was split into two domains based on “relative humidity”. The data with a “relative humidity” of greater than 0.5 were chosen as the source domain, and the remaining data were used to form the target domain. Further, the two attributes in the source data were all perturbed by random numbers following a normal distribution $N(0.1, 0.1)$, and the two attributes in the target data were perturbed by the normal random numbers following $N(7, 1)$ and $N(5, 1)$ respectively. There are 3600 labeled instances in the source domain and 1200 instances in the target data with 15 labeled.

Five-fold cross validation was used for all experiments, and the results are shown in Table 4.9.

Table 4.9 Comparison of the three methods used in real-world datasets

	CCS	HD	ISE	AQ
Q	0.11±0.02	0.11±0.01	0.09±0.04	0.13 ± 0.02
Q_1	1.07±0.16	2.35±0.51	3.05±0.70	6.41 ± 0.35
Q_2	6488.52± 4938.04	5.94± 6.29	76627.04 ± 170157.35	7.27 ± 15.75
Q_3 (method 1)	0.18±0.06	0.60±0.79	0.12±0.00	0.15 ± 0.00
Q_3 (method 2)	0.15±0.01	0.18±0.06	0.13±0.00	0.15 ± 0.01
Q_3 (method 3)	0.91±1.57	0.19±0.10	0.15±0.01	0.15 ± 0.00

From the results, we can see that the mean values of Q_3 in all three methods are all smaller than the mean values for Q_1 and Q_2. This indicates that the models constructed using the proposed methods are better than both the existing source domain model and the model built using few labeled target data. The first and second methods build well-performing models for the target domain using the "concrete compressive strength" dataset. Compared to the first method, the second and third methods did a good job transferring the knowledge from the source domain to the target domain in the "housing" dataset. All the three methods work well in the "Istanbul stock exchange" dataset, and the first method showed a slight lead. On the "air quality" dataset, the three methods showed similar results and all worked well in addressing this domain adaptation problem.

4.5.3 Fuzzy transfer learning for label space adaptation

Sections 4.5.1 and 4.5.2 focus on the fuzzy domain adaptation in regression problems. In this section, the proposed methods are used to some special domain adaptation scenarios in classification task.

Experiment A: Transferring for label space adaptation.

This experiment concentrates on the case where the difference in source and target data is only evident in the label's distributions, thus we apply the second method of changing the output space and the third method of changing both the input and output to handle the adaption in label space. The source and target data are displayed in Figs. 4.7 and 4.8, and are generated following the identical input distributions. The data labeled '0' are represented with blue circle and the data labeled'1' are shown as red asterisk. The distributions of the labels are arranged quite differently in the two domains.

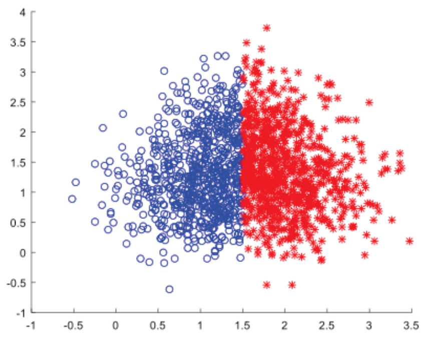


Figure 4.7 Source data for exp A

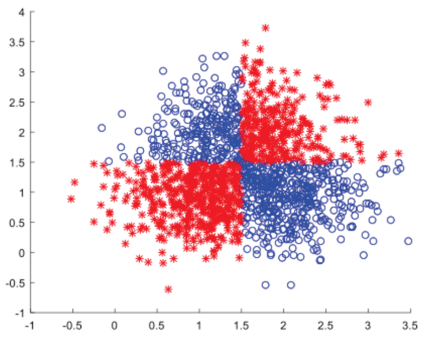


Figure 4.8 Target datta for exp A

In addition, the number of clusters, i.e., the number of fuzzy rules, is an important parameter that is difficult to determine and highly dependent on the dataset. In this case, we adopt different numbers of clusters and compare the results to find the optimal. All the models' construction applies five-fold cross-validation, and the results of Q, Q_1, Q_2 and Q_3 with different number of clusters are shown in Table 4.10 and Table 4.11.

Table 4.10 Results with three clusters (fuzzy rules)

	Q	Q_1	Q_2	Q_3
1	99.67	39.46	81.62	84.26
2	99.00	39.66	81.96	81.35
3	98.67	38.99	70.54	76.35
4	99.00	39.59	71.89	90.88
5	99.33	39.32	53.04	82.57
mean	99.13	39.41	71.81	83.08
stand deviation	0.38	0.27	11.76	5.26

Table 4.11 Results with four clusters (fuzzy rules)

	Q	Q_1	Q_2	Q_3
1	99.00	39.73	68.51	75.68
2	95.33	37.77	58.04	82.16
3	95.00	37.50	59.46	76.42
4	99.33	39.39	66.22	75.07
5	95.00	39.05	50.20	74.59
mean	96.73	38.69	60.49	76.78
stand deviation	2.23	1.00	7.25	3.08

From the results, we can see that the mean value of Q reached 99.13% and 96.73%, and the standard deviation was very small, which means a good model for the source domain has been achieved. However, the mean value for Q_1 was also small; 39.41% for 3 clusters and 38.69% for 4 clusters. This indicates that prediction tasks in the target domain cannot be solved by the source domain's existing model. The mean value for Q_2 was also not very high, and this verifies the assumption of this work: that models constructed with little training data have a high probability of performing badly. Yet, the mean values for Q_3 were always greater than the mean values for Q_1 and Q_2. This indicates that the model trained using the proposed method is superior, not only to the source model but also to the target model trained with few labeled target data.

We noticed that the standard deviations for Q_2 and Q_3 were not small; 11.76% and 5.26% with 3 clusters and 7.25% and 3.08% with 4 clusters. This phenomenon is reasonable and expected, because few labeled target data were available to train the new model – even less was available for the testing data. Obviously, the standard deviation will decline if more labeled target data are available, but this is not consistent with the assumption that the labeled target data is insufficient to build a well-performed model for prediction tasks in the target domain.

Experiment B: Adapting to the change of both input and output.

In the second experiment, we deal with situations where the source data and target data are different in both the input and labels' distributions.

The source data and target data are shown in Figs. 4.9 and 4.10. Similarly, data with disparate labels are represented by points in contrasting colors and marker types, blue circles and red asterisk. The source data shows the same pattern as that in Experiment A, but the target data is quite contrasted.

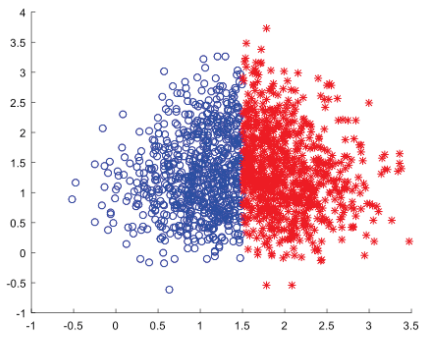


Figure 4.9 Source data for exp. B

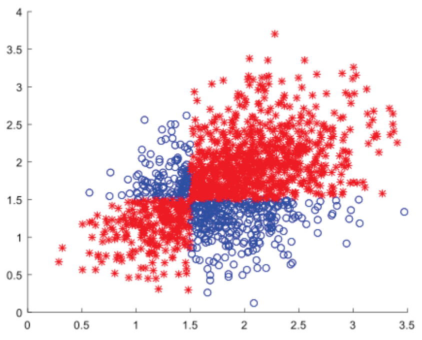


Figure 4.10 Target data for exp. B

Likewise, five-fold-cross validation was applied to build the models. The experimental results are shown in Tables 4.12 and 4.13.

Similarly, the results show that the proposed method improves the performance of the existing source model when classifying the data in the target domain. The standard deviation for Q_3 was not small; 4.77% with 3 clusters and 11.28% with 4 clusters. As with Experiment A, the number of labeled target data was too low. Another possible reason is the large gap between the source data and target data.

Table 4.12 Results with three clusters (fuzzy rules)

	Q	Q_1	Q_2	Q_3
1	99.67	62.30	82.77	79.46
2	99.00	62.70	55.41	88.38
3	98.67	62.30	69.26	90.14
4	99.00	62.64	77.36	81.35
5	99.33	62.09	72.03	81.42
mean	99.13	62.41	71.36	84.15
stand deviation	0.38	0.26	10.32	4.77

Table 4.13 Results with four clusters (fuzzy rules)

	Q	Q_1	Q_2	Q_3
1	99.00	62.64	81.22	68.45
2	95.33	61.28	45.81	74.12
3	95.00	58.85	57.50	85.74
4	99.33	62.50	85.00	86.35
5	95.00	59.86	84.19	60.14
mean	96.73	61.03	70.74	74.96
stand deviation	2.23	1.65	17.96	11.28

Experiment C: Solving the real-world datasets.

A public dataset, called the banknote authentication dataset, from the Machine Learning Repository was used to verify the effectiveness of the proposed methods. The aim of this dataset is to distinguish between genuine and forged 'banknote-like' specimens. The data were obtained by extracting the features of the images of the genuine and forged banknotes using a wavelet transform tool.

The features of this dataset are shown in Table 4.14.

Table 4.14 Attributes of the "banknote authentication" dataset

Feature 1	Variance of wavelet transformed image
Feature 2	Skewness of wavelet transformed image
Feature 3	Curtosis of wavelet transformed image
Feature 4	Entropy of wavelet transformed image

All the features were normalized, and the dataset was split into two datasets based on the "skewness of wavelet transformed image" feature. The data with this feature value larger than 0.5 fell within the source domain; the remaining data were allocated to the target domain. In total, the source domain contained 960 labeled data items in the source domain, but the target domain only contained 30 labeled data items and 320 unlabeled items. Since the input distributions in two domains differ, the method of changing both the input and output is applied to solve this problem. Five-fold cross-validation was used to construct all the models. The experimental results are shown in 4.15.

Table 4.15 Results of the "banknote authentication" dataset

	Q	$Q1$	$Q2$	$Q3$
1	94.79	52.42	55.15	83.64
2	96.35	53.03	40.91	87.88
3	95.83	53.33	68.48	85.45
4	96.88	53.03	69.09	63.94
5	94.27	51.52	53.03	74.24
mean	95.63	52.67	57.33	79.03
stand deviation	1.08	0.72	11.79	9.89

In analyzing the results shown in Table 4.15, it is clear that the mean value for Q_2 was only 52.67%, which indicates the model built using previous data is invalid. The mean value for Q_2 was also small, at 57.33%, and even worse, the standard deviation was not small, at 11.79%. The low accuracy and high standard deviation for Q_2 are testament to our assumption in this work: that a small amount of labeled data cannot guarantee the performance of the constructed models. The performance of the proposed method is reflected in the values for Q_3, which show a mean value of 79.03% and a standard deviation of 9.89%. The mean value for Q_3 was not greater than that of Q_1 and Q_2. This suggests the proposed model is superior to both the previous model and the model built only with target data.

4.6 Summary

In this chapter, we propose three granular fuzzy regression domain adaptation methods to address three challenging cases in fuzzy domain adaptation: where the conditions, conclusions, or both the conditions and conclusions of the fuzzy rules in the source and targets domains differ. These methods modify the input and/or output of the data space through mappings to make the fuzzy rules of the existing model more compatible for solving tasks in the target domain. Our methods effectively solve regression problems in the target domain even when only a small amount of labeled data is available. More importantly, these three methods constitute an entire framework to provide guidance for the domain adaptation in TS fuzzy model. The experiments on the synthetic and real-world datasets demonstrate the capability of the proposed methods in dealing with the fuzzy domain adaptation in regression prediction task. Moreover, the presented method are used to solve a specific case in the classification problem, where the label space has shifted in the target domain. The experimental results show that the method of changing the output space can handle the discrepancy of label space in two domains very well.

Chapter 5

Fuzzy Domain Adaptation for the Dismatch of Fuzzy Rules

5.1 Introduction

This chapter concentrates on the homogeneous domain adaptation problems where the two domains have a different number of fuzzy rules. Both situations are addressed: where the number of fuzzy rules in the source domain is greater than in the target domain, and vice versa.

When the source domain has the greater number of fuzzy rules, the source model can still be used and modified to fit the target data because TS fuzzy models weight some linear functions in a nonlinear way when fitting to a curve. Each linear function represents a fuzzy rule, and all the fuzzy rules form a fuzzy partition of the output space. The greater the number of fuzzy rules, the more precise the partition of the space is. It is, therefore, reasonable to consider that a TS fuzzy model can approximate any curve as long as there is an adequate number of fuzzy rules. And, if there is a greater number of fuzzy rules in the source domain than in the target domain, then it is reasonable to revise the fuzzy rules in the source domain to fit the target data.

When the target domain has the greater number of fuzzy rules, the fuzzy rules in the source domain must be reconstructed to a number that is no less than the target domain. Then, the rebuilt fuzzy rules are modified and applied to fit the target data.

Further, in addition to target data with labels, target data without labels are also used to improve the performance of the target model. We assume that instances that are close to each other in the input space will have similar labels. The closer the instances, the more similar their labels are.

The rest of this chapter is organized as follows. Section 5.2 states the problem we aim to cope with. Section 5.3 describes the proposed methods and the corresponding optimization procedure. The experiments on Section 5.4 validate the proposed methods, and compare two ways of constructing mappings for the input space. Moreover, the presented method is used the handle the transfer learning in classification problems. Finally, summary of this chapter is discussed in Section 5.5.

5.2 Problem statement

The dataset in the source domain is denoted by $\boldsymbol{D} = \{(\boldsymbol{x}_1^s, y_1^s), \cdots, (\boldsymbol{x}_{N_s}^s, y_{N_s}^s)\}$, where $\boldsymbol{x}_k^s \in R^n$, $k = 1, \cdots, N_s$, is the n-dimensional input variable, the label $y_k^s \in R$ is the continuous output variable, and N_s indicates the number of data. Since the amount of source data with labels is massive, a well-performing regression model for the source domain – the TS fuzzy model M^s – can be learned.

model M^s

$$\text{if } \boldsymbol{x}^s \text{ is } A_i(\boldsymbol{x}^s,\ \boldsymbol{v}_i^s),\ \text{then } y^s \text{ is } L_i(\boldsymbol{x}^s, \boldsymbol{a}_i^s) \qquad i = 1, 2, ..., c_s \tag{5.1}$$

The dataset in the target domain contains two subsets: one with labels and one without labels $\boldsymbol{H} = \{\boldsymbol{H}_L, \boldsymbol{H}_U\} = \{\{(\boldsymbol{x}_1^t, y_1^t), \cdots, (\boldsymbol{x}_{N_{t1}}^t, y_{N_{t1}}^t)\}, \{\boldsymbol{x}_{N_{t1+1}}^t, \cdots, \boldsymbol{x}_{N_t}^t\}\}$, where $\boldsymbol{x}_k^t \in R^n$, $k = 1, \cdots, N_t$ is the n_t-dimensional input variable, $y_k^t \in R$, $k = 1, \cdots, N_{t1}$ is the continuous output variable. $\boldsymbol{H}_L$ includes the instances with labels, and $\boldsymbol{H}_U$ contains the data without labels. The numbers of data in $\boldsymbol{H}_L$ and $\boldsymbol{H}_U$ are N_{t1} and $N_t - N_{t1}$ respectively, and satisfy $N_{t1} << N_t$, $N_{t1} << N_s$.

Suppose the ideal model for the target domain is M^t.

model M^t

$$\text{if } \boldsymbol{x}^t \text{ is } A_i(\boldsymbol{x}^t,\ \boldsymbol{v}_i^t),\ \text{then } y^t \text{ is } L_i(\boldsymbol{x}^t, \boldsymbol{a}_i^t) \qquad i = 1, 2, ..., c_t \tag{5.2}$$

Different with the problem in Chapter 4, the numbers of fuzzy rules in two domains are different, which impedes the usage of source model on the target data.

5.3 Fuzzy transfer learning for dismatch of fuzzy rules

Our **f**uzzy **ho**mogeneous **d**omain **a**daptation (FHoDA) method for dealing with knowledge transfer with fuzzy rule-based models follows. The FHoDA method contains two main steps.

Step 1: Train the source model.

A source model M^s is built based on the source dataset $\boldsymbol{D}$.

model M^s

$$\text{if } \boldsymbol{x}_k^s \text{ is } A_i(\boldsymbol{x}_k^s,\ \boldsymbol{v}_i^s),\ \text{then } y_k^s \text{ is } L_i(\boldsymbol{x}_k^s, \boldsymbol{a}_i^s) \qquad i = 1, 2, \cdots, c \tag{5.3}$$

where $c = \max(c_s, c_t)$.

$c = \max(c_s, c_t)$ ensures that the number of trained fuzzy rules are sufficient to fit the target data later. So, in the first situation, c is simply equal to c_s. But in the second situation, c is equal to c_t, which means that the source model is reconstructed with the same number of fuzzy rules as the target domain. This can facilitate modification and transfer of the fuzzy rules between domains.

Step 2: Modify the existing fuzzy rules to construct the target model.

There can be two possible differences between the fuzzy rules in source and target domains: their conditions and/or their conclusions. However, because the number of fuzzy rules is not equal, it is hard to tell exactly where the differences are – the conditions or the conclusions. To guarantee the best results, we modify the existing model using the three algorithms proposed in our previous paper Zuo et al. (2017b) and select the one with the best performance as the target model, i.e., we change the input space, change the output space, and change both spaces.

The corresponding target models are:

model $\overline{M}_1^t$ (changes of input space)

$$\text{if } \boldsymbol{x}_k^t \text{ is } A_i(\Phi(\boldsymbol{x}_k^t),\ \boldsymbol{v}_i^s),\ \text{then } y_k^t \text{ is } L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s) \qquad i = 1, 2, \cdots, c \tag{5.4}$$

model $\overline{M}_2^t$ (changes of output space)

$$\text{if } \boldsymbol{x}_k^t \text{ is } A_i(\boldsymbol{x}_k^t,\ \boldsymbol{v}_i^s),\ \text{then } y_k^t \text{ is } \Psi_i(L_i(\boldsymbol{x}_k^t, \boldsymbol{a}_i^s)) \qquad i = 1, 2, \cdots, c \tag{5.5}$$

model $\overline{M}_3^t$ (changes of input and output spaces)

$$\text{if } \boldsymbol{x}_k^t \text{ is } A_i(\Phi(\boldsymbol{x}_k^t),\ \boldsymbol{v}_i^s), \text{ then } y_k^t \text{ is } \Psi_i(L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s)) \qquad i = 1, 2, \cdots, c \tag{5.6}$$

where $\Phi = [\Phi_1 \cdots \Phi_n]$, $n = n_s = n_t$, and $\Psi = [\Psi_1 \cdots \Psi_c]$ are the transformation mappings for the input space and output space.

The final target model $\overline{M}^t$ is chosen from the best among models $\overline{M}_1^t$, $\overline{M}_2^t$ and $\overline{M}_3^t$, i.e.,

$$\overline{M}^t = \overline{M}_i^t, \text{if } \overline{M}_i^t \geq \overline{M}_j^t,\ i, j = 1.2.3 \tag{5.7}$$

where $\hat{M}_i^t \geq \hat{M}_j^t$ means the performance of $\hat{M}_i^t$ on the target data $\boldsymbol{H}_U$ is no worse than that of $\hat{M}_j^t$.

The construction of mappings Φ and Ψ is the key element in the FHoDA method. The nonlinear functions are used to build the mappings Zuo et al. (2017b).

Apart from the nonlinear transformation, we also construct the mappings of input variables using the piecewise linear functions with the structure shown in Fig. 5.1.

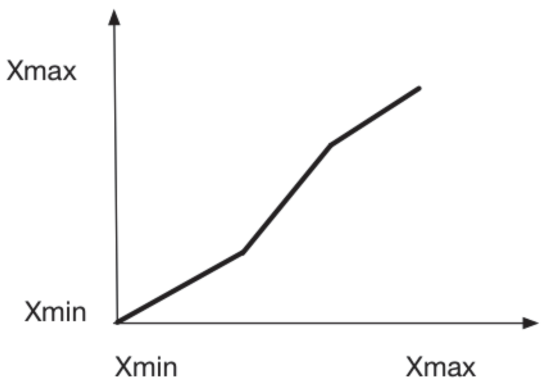


Figure 5.1 Architecture of piecewise linear mapping

We construct a piecewise linear mapping for each input variable, which is a monotonous function defined on the domain of the input variable.

$$\Phi\colon\ [\boldsymbol{x}^t_{min},\boldsymbol{x}^t_{max}] \rightarrow [\boldsymbol{x}^t_{min},\boldsymbol{x}^t_{max}] \tag{5.8}$$

where $\boldsymbol{x}^t_{min}$ and $\boldsymbol{x}^t_{max}$ are the minimum and maximum values of input variable $\boldsymbol{x}^s$. So the endpoints of the piecewise linear function Φ are $(\boldsymbol{x}_{min},\boldsymbol{x}_{min})^t$ and $(\boldsymbol{x}^t_{max},\boldsymbol{x}^t_{max})$.

Suppose there are two cutoff points in the piecewise linear function, and then the mapping Φ is written as:

$$\Phi(x^t_{kj}) = \begin{cases} \frac{b_1 - x^t_{mj}}{\alpha_1 - x^t_{mj}} * x^t_{kj} + \frac{x^t_{mj} * a_1 - x^t_{mj} * b_1}{\alpha_1 - x^t_{mj}}, x^t_{mj} \leq x^t_{kj} < a_1 \\ \\ \frac{b_2 - b_1}{a_2 - a_1} * x^t_{kj} + \frac{b_1 * a_2 - a_1 * b_2}{a_2 - a_1}, a_1 \leq x^t_{kj} < a_2 \\ \\ \frac{x^t_{Mj} - b_2}{x^t_{Mj} - a_2} * x^t_{kj} + \frac{b_2 * x^t_{Mj} - a_2 * x^t_{kj}}{x^t_{Mj} - a_2}, a_2 \leq x^t_{kj} \leq x^t_{Mj} \end{cases} \tag{5.9}$$

where (a_1, b_1) and (a_2, b_2) are the two cutoff points of the piecewise linear function.

The parameters of the mappings Φ and Ψ are obtained through an optimization procedure, but the cost functions are different when optimizing Φ and Ψ to get models $\overline{M}^t_1$, $\overline{M}^t_2$ and $\overline{M}^t_3$.

When training model $\overline{M}^t_1$, i.e., applying the algorithm that changes the input space, the cost function is:

$$S1 = \sqrt{\frac{1}{N_{t1}} \sum_{k=1}^{N_{t1}} (\sum_{i=1}^{c} \frac{A_i(\Phi(\boldsymbol{x}^t_k), \Phi(\boldsymbol{v}^s_i))}{\sum_{j=1}^{c} A_j(\Phi(\boldsymbol{x}^t_k), \Phi(\boldsymbol{v}^s_j))} L_i(\Phi(\boldsymbol{x}^t_k), \boldsymbol{a}^s_i) - y^t_k)^2 +}$$
$$\lambda_1 \sqrt{\frac{1}{N_{t1} * h} \sum_{k=1}^{N_{t1}} \sum_{l=1}^{h} (y^t_k - y^t_k(l))^2 * \exp(-\|x^t_k - x^t_k(l)\|)} + \frac{\lambda_2}{2} w^T w \tag{5.10}$$

where $y_k^t(l) = \sum_{i=1}^{c} \frac{A_i(\Phi(x_k^t(l)),\Phi(\boldsymbol{v}_i^s))}{\sum_{j=1}^{c} A_i(\Phi(x_k^t(l)),\Phi(\boldsymbol{v}_j^s))} L_i(\Phi(x_k^t(l)),\boldsymbol{a}_i^s)$.

The first term in cost function 5.10 trains the model based on the target data with labels, which aims to minimize the gap between the output of the constructed model and the target data's real response. The second term operates on the assumption that data with less distance in the input space will have a similar response. Therefore, for each target data $\boldsymbol{x}_k^t$ in $\boldsymbol{H}_L$, the h-nearest data $\{\boldsymbol{x}_k^t(1)\cdots\boldsymbol{x}_k^t(h)\}$ in $\boldsymbol{H}_U$ are found, and the outputs of $\{\boldsymbol{x}_k^t(1)\cdots\boldsymbol{x}_k^t(h)\}$ are expected to be close to the response of $\boldsymbol{x}_k^t$. $\exp(-\|\boldsymbol{x}_k^t - \boldsymbol{x}_k^t(l)\|$ determines that the data that are closer to the center $\boldsymbol{x}_k^t$, will have an output more approximate to the response of $\boldsymbol{x}_k^t$. The third term is a structural risk of the cost function, and parameter λ_2 indicates the tradeoff between the quality of an approximation and the complexity of the approximation function. w is the vector of all the optimized parameters.

When training model $\overline{M}_2^t$, i.e., applying the algorithm that changes the output space, the cost function is:

$$S2 = \sqrt{\frac{1}{N_{t1}} \sum_{k=1}^{N_{t1}} (\sum_{i=1}^{c} \frac{A_i(\boldsymbol{x}_k^t, \boldsymbol{v}_i^s)}{\sum_{j=1}^{c} A_j(\boldsymbol{x}_k^t, \boldsymbol{v}_j^s)} \Psi_i(L_i(\boldsymbol{x}_k^t, \boldsymbol{a}_i^s)) - y_k^t)^2} + \frac{\lambda_2}{2} w^T w \qquad (5.11)$$

The cost function for training the mappings for the output space contains two terms. Both are the same as the first and third term in $S1$ – one trains the model with the target data with labels, the other restrains the complexity of the approximation function. But here, we do not use target data without labels to train the target model. This is because $\boldsymbol{x}_k^t$ needs to find the h -nearest data based on distance in the input space to best use data without labels in the target domain. However, the algorithm that changes the output space focuses on modifying the output variables; therefore, using the data without labels here may have a negative impact on the model's construction.

When training model $\overline{M}_3^t$, i.e., applying the algorithm that changes the input and output spaces, the cost function is:

$$S3 = \sqrt{\frac{1}{N_{t1}} \sum_{k=1}^{N_{t1}} \left(\sum_{i=1}^{c} \frac{A_i(\Phi(\boldsymbol{x}_k^t), \Phi(\boldsymbol{v}_i^s))}{\sum_{j=1}^{c} A_j(\Phi(\boldsymbol{x}_k^t), \Phi(\boldsymbol{v}_j^s))} \Psi_i(L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s)) - y_k^t \right)^2} + \frac{\lambda_2}{2} w^T w \tag{5.12}$$

Similarly, only the target data with labels are applied to train the mappings for the input and output spaces simultaneously. And a structural risk is added to the cost function to control the complexity of the approximation function.

The overall algorithm for the FHoDA method described above is provided in Algorithm 1.

Algorithm 1. Homogeneous domain adaptation procedure

```
Input: $\boldsymbol{D}$, $\boldsymbol{H}$,
Output: $\boldsymbol{Y}_U$ for $\boldsymbol{H}_U$
1. Train source model $M^s$ based on $\boldsymbol{D}$
2. Modify the fuzzy rules in $M^s$
    2.1 Change input space, and get model $M_1^t$
    2.2 Change output space, and get model $M_2^t$
    2.3 Change both input and output spaces, and get model $M_3^t$
3. Compare $M_1^t$, $M_2^t$, and $M_3^t$, and choose the best one as $M^t$
4. Use $M^t$ to predict the response $\boldsymbol{Y}_U$ for $\boldsymbol{H}_U$
```

5.4 Empirical results analysis

The experiments in this section comprise three sections. The first section introduces the synthetic datasets and the experimental settings. The second section validates the effectiveness of the proposed FHoDA method. The final section analyzes the sensitivity of some critical parameters.

5.4.1 Datasets and experimental settings

Four synthetic datasets were generated with different numbers of fuzzy rules to simulate various cases of homogeneous domain adaptation. As shown in Table 5.1, 500 instances were generated for each rule, so dataset 2r contained 1000 instances, dataset 3r contained 1500, and so on.

Table 5.1 Four datasets with a different number of fuzzy rules

	number of fuzzy rules	number of instances
dataset 2r	2	1000
dataset 3r	3	1500
dataset 4r	4	2000
dataset 5r	5	2500

In each experiment, two of the four datasets were chosen as the source and the target domain respectively. All the data in the source domain had labels, but only 5% of the data in the target domain had labels. The remaining labels were only available during the testing procedure. In total, 12 experiments simulating homogeneous domain adaptation with the fuzzy rule-based models were executed.

The FHoDA method generates the target model through an optimization process. In this work, a DE optimization algorithm was used to optimize the parameters of the constructed mappings. DE is a computational method that determines an optimal solution by iteratively navigating a population of solutions to minimize a certain predetermined objective function. Such algorithms are commonly known as metaheuristics, as they make few, or no, assumptions about the problem being optimized and are able to search very large populations of candidate solutions. Beyond DE algorithms, PSO algorithms were also frequently used. However, based on the experimental results from our previous studies Zuo et al. (2017a), their performance and stability on this class of problems are inferior, so we chose a DE algorithm to optimize the parameters of the transformation mappings and construct the target model. Five-fold cross validation was used for the construction, so all results are shown in the form "mean$\pm$variance".

5.4.2 Validation of algorithm effectiveness

As described above, 12 experiments simulating homogeneous domain adaptation with fuzzy rule-base models were conducted. The results are shown in Table 5.2.

The left column in Table 5.2 indicates the source and target domains. For example, '5r to 4r' indicates that the source domain is 'dataset 5r', and the target domain is 'dataset 4r'. The second column shows the RMSE of the source model on the target data without labels $\boldsymbol{H}_U$, which is treated as the baseline of the transfer learning problem. The results in columns 3-5 are the RMSE of the models $\overline{M}_1^t$, $\overline{M}_2^t$ and $\overline{M}_3^t$, on target data $\boldsymbol{H}_U$. The model with the best performance among $\overline{M}_1^t$, $\overline{M}_2^t$ and $\overline{M}_3^t$ was then selected as the final target model $\overline{M}^t$. The last column displays the RMSE of $\overline{M}^t$ on $\boldsymbol{H}_U$, which is simply the minimum value from columns 3-5.

Comparing the values in the second and sixth columns of Table 5.2, we can conclude that the proposed FHoDA method efficiently transfers knowledge between the domains and greatly improves the ability of the modified model to solve regression tasks in the target domain. In addition to validating the effectiveness of the proposed method, we also analyzed the performance of models $\overline{M}_1^t$, $\overline{M}_2^t$ and $\overline{M}_3^t$. In nine of the twelve experiments, model $\overline{M}_2^t$ – the algorithm that changes the output space – outperformed the other two models, $\overline{M}_1^t$, $\overline{M}_2^t$ and $\overline{M}_3^t$, the algorithm that changes the input space, provided an advantage in only three experiments, while model $\overline{M}_3^t$, $\overline{M}_2^t$ and $\overline{M}_3^t$, which changes both spaces, was inferior in all 12 experiments. That said, we cannot assert that $\overline{M}_3^t$ will always fail to compete because the performance of these three models heavily depends on the dataset. The impracticality of experimenting with all, or at least enough, datasets means we need to retain an algorithm that changes both the input and output spaces in our method for the foreseeable future.

Moreover, since target data without labels $\boldsymbol{H}_U$ are used to improve model $\overline{M}_1^t$'s performance, Table 5.3 compares the RMSE of model $\overline{M}_1^t$ when built with and without target data that had no labels. Lower values are shown in bold.

The results show that using target data $\boldsymbol{H}_U$ for training was better in ten of the twelve experiments – a clear performance improvement for $\overline{M}_1^t$. However, in two experiments, using target data without labels had a negative impact. The cause may lie in the model's construction. If some of the target data with labels were located at the junction of two clusters (fuzzy rules), the utilization of $\boldsymbol{H}_U$, finding the h-nearest data without labels for each target data with response, will result in an inappropriately constructed model.

Table 5.2 Results of the twelve experiments

Source to target	RMSE of the models				
	M^s	$\overline{M}_1^t$	$\overline{M}_2^t$	$\overline{M}_3^t$	$\overline{M}^t$
5r to 4r	5.0019± 0.0000	1.0756± 0.0004	0.6221± 0.0119	1.0207± 0.0253	0.6221± 0.0119
5r to 3r	4.1156± 0.0000	0.8962± 0.0083	0.9803± 0.0553	1.4873± 0.0553	0.8962± 0.0083
5r to 2r	5.5329± 0.0000	0.5573± 0.0005	1.0150± 0.3262	1.7513± 0.7348	0.5573± 0.0005
4r to 3r	1.1917± 0.0000	2.0996± 0.1718	0.6364± 0.0211	2.2208± 0.6394	0.6364± 0.0211
4r to 2r	2.2465± 0.0000	1.5711± 0.0108	0.9608± 0.0855	1.2663± 0.3276	0.9608± 0.0855
3r to 2r	2.3964± 0.0000	0.5772± 0.0005	0.6980± 0.1221	1.0758± 0.0067	0.5772± 0.0005
2r to 3r	2.2852± 0.0000	0.8074± 0.0044	0.6862± 0.0036	0.8522± 0.0061	0.6862± 0.0036
2r to 4r	2.3701± 0.0002	1.2947± 0.0039	0.6618± 0.0031	0.7212± 0.0017	0.6618± 0.0031
2r to 5r	6.4372± 0.0000	3.7228± 0.0002	1.0628± 0.0245	3.2088± 4.4997	1.0628± 0.0245
3r to 4r	1.0709± 0.0002	0.8457± 0.0005	0.6173± 0.0024	0.9669± 0.0045	0.6173± 0.0024
3r to 5r	4.5296± 0.0005	2.5397± 0.0014	1.4062± 0.0073	1.9163± 0.0726	1.4062± 0.0073
4r to 5r	4.2761± 0.0165	3.0614± 0.0037	1.3590± 0.0026	2.3501± 0.0070	1.3590± 0.0026

Table 5.3 $\overline{M}_1^t$ built using/not using $\boldsymbol{H}_U$ - model comparison

Source to target datasets	RMSE of the models	
	model $\overline{M}_1^t$ (not using $\boldsymbol{H}_U$)	model $\overline{M}_1^t$ (using $\boldsymbol{H}_U$)
5r to 4r	1.0781± 0.0004	**1.0756± 0.0004**
5r to 3r	0.9352± 0.0057	**0.8962± 0.0083**
5r to 2r	0.5602± 0.0016	**0.5573± 0.0005**
4r to 3r	2.1269± 0.2059	**2.0996± 0.1718**
4r to 2r	1.5981± 0.0310	**1.5711± 0.0108**
3r to 2r	0.5882± 0.0016	**0.5772± 0.0005**
2r to 3r	0.8080± 0.0093	**0.8074± 0.0044**
2r to 4r	1.2522± 0.0009	1.2947± 0.0039
2r to 5r	3.6904± 0.0012	3.7228± 0.0002
3r to 4r	0.8876± 0.0009	**0.8457± 0.0005**
3r to 5r	2.5273± 0.0007	**2.5397± 0.0014**
4r to 5r	3.0755± 0.0110	**3.0614± 0.0037**

5.4.3 Parameter sensitivity analysis

Within FHoDA's optimization procedure, some parameters play an important role in model construction in the target domain. The three groups of experiments, shown in Figs. 5.2 - 5.4, were designed to explore the impact of these parameters. Fig. 5.2 depicts the effect of parameter p (the number of nodes used to construct mappings for the input space) on model $\hat{M}_1^t$'s performance. Fig. 5.3 shows model $\overline{M}_2^t$'s performance with a varying q (the number of nodes used when building the mappings for the output space). Fig. 5.4 charts model $\hat{M}_1^t$'s performance with different values for h (the number of target data without labels selected for each target data with response). Only the results for the experiments '5r to 4r', '5r to 3r' and '5r to 2r' have been included in the figures to illustrate the impact of the parameters on the performance of the model.

Observing the results shown in Fig. 5.2, the parameter p had a slight impact on model construction in experiment '5r to 2r'. The variations in the model's performance as p varied were almost the same in experiments '5r to 4r' and '5r to 3r'. When p was changed from 2 to 4, the RMSE increased, then decreased with a p of 5, and peaked at a p of 6.

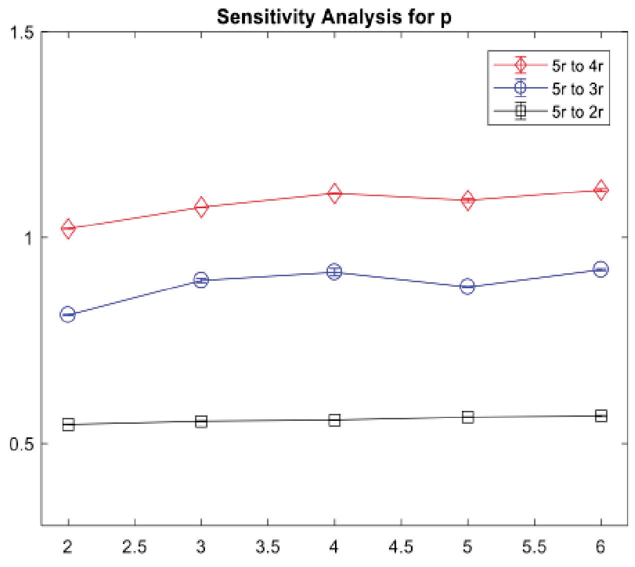


Figure 5.2 Sensitivity analysis of parameter p in three experiments

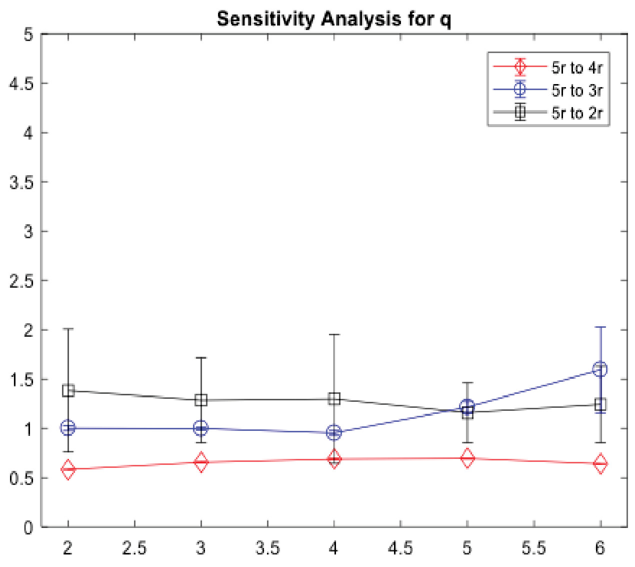


Figure 5.3 Sensitivity analysis of parameter q in three experiments

The results in Fig. 5.3 show no obvious changes to model construction with different values of q in the '5r to 4r' experiment. In '5r to 3r', the RMSE shows a rising trend at q values greater than 4, and, in '5r to 2r', the RMSE fluctuates as the q values change.

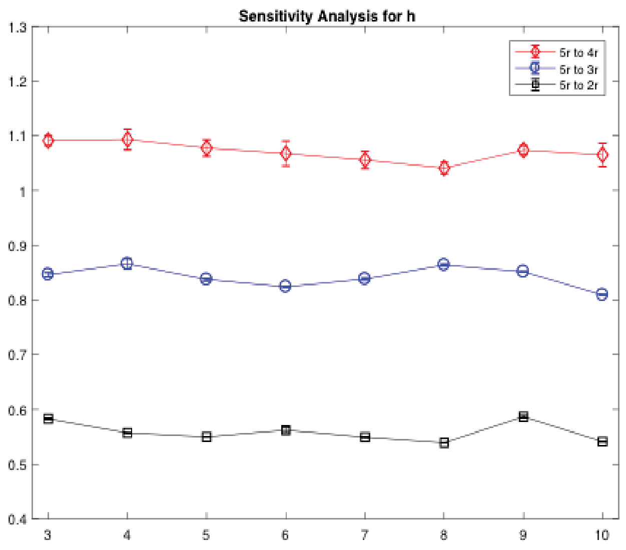


Figure 5.4 Sensitivity analysis of parameter h in three experiments

Fig. 5.4 shows fluctuations in all three experiments. The model's best performance appears when h=8 in '5r to 4r', h =10 in '5r to 3r', and h =8 in '5r to 2r'. Notice that the performance of the model does not always develop an increasing trend. When more data without labels around the target data with labels are selected to improve model construction, data at greater distances are found. It is unreasonable to suppose that data at large distances will have similar labels, so a growing number of selected target data without labels will eventually lead to a negative impact on the model's optimization.

5.5 Comparison of two ways of constructing mappings

Two experiments are executed to compare the two ways of constructing the mappings for the input space, the nonlinear functions and the piecewise linear functions.

Experiment A: Data with obvious partitions.

In this experiment, we utilize a source dataset that has obvious partitions. The source data (left) and target data are shown in Fig. 5.5 and Fig. 5.6.

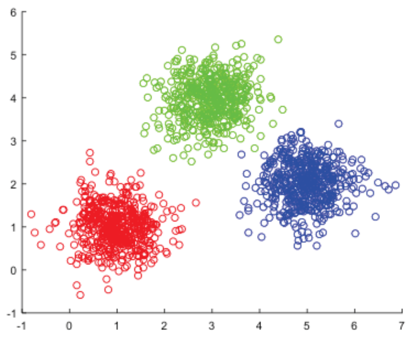


Figure 5.5 Source data

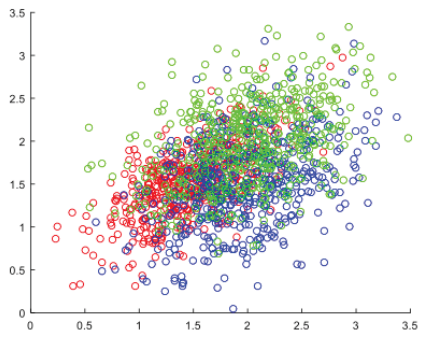


Figure 5.6 Target data

There are 500*3 instances in both source domain and target domain. But in target domain, only 1% data are as training set, and the remaining 99% data are as testing data. We can see that the distributions of source data and target data are quite different. The partition in source domain is quite obvious, but there are no obvious bounds between the clusters in target domain. Therefore, if there are only few data in target domain, then they can't guarantee the accuracy of the trained model, and the performance of the built model heavily depends on the positions of the training data. Especially, if the training data locate in the cross-section of the clusters, then the accuracy of the trained model in testing model maybe quite bad.

Here, nonlinear continuous functions based on sigmoid functions and piece-wise linear functions are used respectively to build the mappings for input variables. Some important parameters in model $\overline{M}^t$ are shown in Table 5.4 and Table 5.5.

Table 5.4 Parameters in model $\overline{M}^t$ (nonlinear mappings)

Fuzzification coefficient in FCM	$m = 1.2$
Number of sigmoid functions in the nonlinear mappings	$P = 5$
The lower and upper bounds of optimized parameters	$\alpha \in [0, 1]$
	$\beta \in [-5, 5]$
	$\omega \in [-5, 5]$

PSO is used here to get the optimal parameters α, β, and ω of mappings Φ. And in PSO, the given ranges of parameters are quite important, because they determine the researching spaces.

Table 5.5 Parameters in model M^t (piecewise linear mappings)

Fuzzification coefficient in FCM	$m = 1.2$
Number of cutoff points in piecewise linear functions	$\mathrm{q} = 4$

q is a parameter that control the shape of the piecewise linear functions.

The experimental results are shown in Table 5.6.

Table 5.6 The regression results

Q	0.0000
Q_1	17.8208
Q_2	11.6251
Q_3(Nonlinear mappings)	1.4424
Q_3(Piecewise linear mappings)	2.7590

Since the target data do not have obvious partitions, the few data in target domain can't train a well-performed model $\overline{M}^s$, which has a high mean square error in testing data ($Q_2 = 11.6251$). Our proposed new method can build a good model for target domain, using the nonlinear mappings or piecewise linear mappings.

Apart from the effectiveness of the proposed method, we are quite care about the mappings of the input variables. Figs. 5.7 and 5.8 and show the nonlinear mappings and piecewise linear mappings for input variable 1 (red) and input variable 2 (blue).

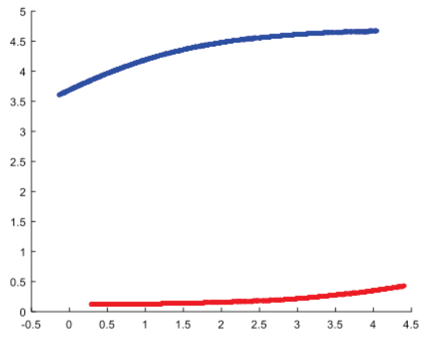


Figure 5.7 Nonlinear mappings

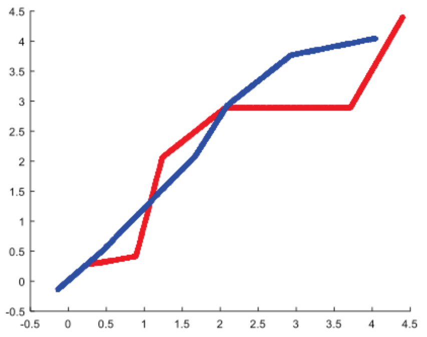


Figure 5.8 Piecewise liner mappings

Experiment B: Data without obvious partitions.

In this experiment, the data in source domain doesn't have obvious partitions. Source data (left) and target data (right) are draw in Fig. 5.9 and Fig. 5.10 respectively.

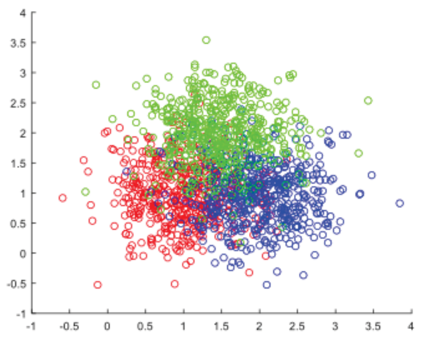


Figure 5.9 Source data

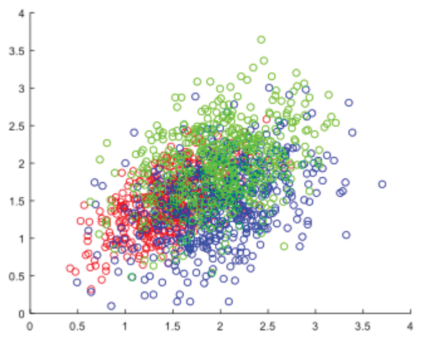


Figure 5.10 Target data

From Fig. 5.9 we could see that there have some crossed regions in the source data. The setting of the parameters in this experiment is the same with that in Experiment A, shown in Tables 5.4 and 5.5. The experimental results are shown in Table 5.7.

Table 5.7 The regression results

Q	0.0068
Q_1	4.5450
Q_2	3605.0528
Q_3(Nonlinear mappings)	2.0409
Q_3(Piecewise linear mappings)	3.0962

Because the clusters in source domain have some overlapping regions, the mean square error of model M^s in source domain is 0.0068 that isn't like the result in Experiment A, close to zero. The model $\overline{M}^t$ has a big error when predicting the value in testing data, which indicates that the data in training set can't identify the partitions of target data. Since in the experiment setting, the 15 training data are randomly selected from the whole target dataset (randomly select 5 instances in every cluster), the training set in two experiments maybe quite different, and this leads to the different values of Q_2 in two experiments. Although the training data in target domain are not quality, the proposed method can still use them to train good mappings for the input variables, and the resulting model M^t has a low error in testing set, 2.0409 and 3.0962 respectively when nonlinear mappings and piecewise linear mappings are constructed.

The mappings of input variable 1 (red) and input variable 2 (blue) are shown in Figs. 5.11 and 5.12.

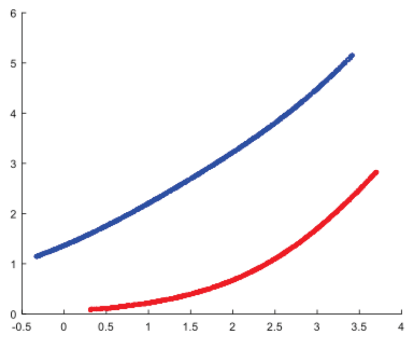


Figure 5.11 Nonlinear mappings

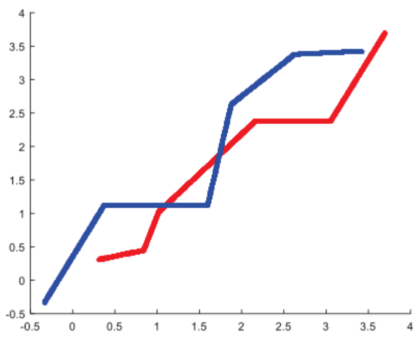


Figure 5.12 Piecewise linear mappings

5.6 Application to the classification problems

The experiments in this subsection include two parts that are distinguished by different relationships between source domain and target domain. Firstly, we will consider the situation that the distributions of source data and target data are different. In addition, since in our experiments all the datasets follow Gaussian distributions that are determined by mean values and variances, the experiments in the first part are also divided into three subparts to explore the validity of the proposed method to address different situations in transfer learning. The second part focuses on the situation that the patterns of classes in two domains have changed. In this section, only the method of changing the input space are implemented to modify the existing models and construct models for target domain.

Experiment A: The distributions of datasets in two domains change

Here, we use the 2-D synthetic data, and consider the following three discrepancies of distributions in two domains:

Table 5.8 Three cases of different distributions

Case 1	The mean values are different, and the variances are the same.
Case 2	The mean values are the same, and the variances are different.
Case 3	The mean values are different, and the variances are different.

The identical fuzzy transfer learning model is constructed for the three cases, and the parameters in model M^t are listed in Table 5.9.

Table 5.9 Parameters in model $\overline{M}^t$ (nonlinear mappings)

fuzzification coefficient in FCM	$m = 1.2$
number of sigmoid functions in the nonlinear mappings	$P = 5$
The lower and upper bounds of optimized parameters	$\alpha \in [0,1]$
	$\beta \in [-5,5]$
	$\omega \in [-5,5]$

In the first case, the mean values of datasets in source and target domains are different, but the variances are the same. Table 5.10 gives the information of datasets in source domain and target domain.

Table 5.10 Information of datasets in source and target domains

Source data			Target Data		
1	$\mu_1^s = [1\ 1]$	$\sigma_1^s = \begin{pmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{pmatrix}$	1	$\mu_1^t = [2\ 1]$	$\sigma_1^t = \begin{pmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{pmatrix}$
2	$\mu_2^s = [1\ 3]$	$\sigma_2^s = \begin{pmatrix} 0.4^2 & 0 \\ 0 & 0.4^2 \end{pmatrix}$	2	$\mu_2^t = [5\ 3]$	$\sigma_2^t = \begin{pmatrix} 0.4^2 & 0 \\ 0 & 0.4^2 \end{pmatrix}$
3	$\mu_3^s = [3\ 2]$	$\sigma_3^s = \begin{pmatrix} 0.3^2 & 0 \\ 0 & 0.3^2 \end{pmatrix}$	3	$\mu_3^t = [2\ 4]$	$\sigma_3^t = \begin{pmatrix} 0.3^2 & 0 \\ 0 & 0.3^2 \end{pmatrix}$

From Table 5.10 we can see that datasets from two domains are labelled according to the distributions, so data with different labels (1, 2, and 3) follow different distributions.

The results of this experiment are listed in Table 5.11.

Table 5.11 Accuracy of models in predicting the labels in domains

Accuracy of model M^s in source domain	98.87%
Accuracy of model M^s in target domain	38.20%
Accuracy of model $\overline{M}^t$ in target domain	80.20%

Because the mean values of datasets in two domains are different, model M^s trained using source data doesn't have a good performance in classifying the data in target domain, the accuracy is only 38.20%. Our method can improve the accuracy of classifying target data, and the accuracy of the new model $\overline{M}^t$ is 80.20% in target domain.

In the second case, the mean values of data in two domains are the same, but the variances are different. Table 5.12 gives the experiments setting of the second case.

Table 5.12 Information of datasets in two domains

Source Data		Target Data	
1	$\boldsymbol{\mu}_1^s = [1\ 1]$ $\boldsymbol{\sigma}_1^s = \begin{pmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{pmatrix}$	1	$\boldsymbol{\mu}_1^t = [1\ 1]$ $\boldsymbol{\sigma}_1^t = \begin{pmatrix} 0.4^2 & 0.1 \\ 0.1 & 0.4^2 \end{pmatrix}$
2	$\boldsymbol{\mu}_2^s = [1\ 3]$ $\boldsymbol{\sigma}_2^s = \begin{pmatrix} 0.4^2 & 0 \\ 0 & 0.4^2 \end{pmatrix}$	2	$\boldsymbol{\mu}_2^t = [1\ 3]$ $\boldsymbol{\sigma}_2^t = \begin{pmatrix} 0.5^2 & 0.1 \\ 0.1 & 0.5^2 \end{pmatrix}$
3	$\boldsymbol{\mu}_3^s = [3\ 2]$ $\boldsymbol{\sigma}_3^s = \begin{pmatrix} 0.3^2 & 0 \\ 0 & 0.3^2 \end{pmatrix}$	3	$\boldsymbol{\mu}_3^t = [3\ 2]$ $\boldsymbol{\sigma}_3^t = \begin{pmatrix} 0.4^2 & 0.1 \\ 0.1 & 0.4^2 \end{pmatrix}$

The data are still labeled based on the distributions. Although the coverage of data in two domains is different, partitions of data in different classes are still very obvious. Therefore the classifier trained using source data will have a good performance in classify the data in target domain, and this has been verified by the experiment result. The accuracy of the existing model in predicting the labels of source data is 98.47%, and in predicting the labels of data in target domain is 97.33% that means the existing model can address the classification tasks in target domain very well.

Finally, we consider the third case that the mean values and variances of data in two domains are both different. Table 5.13 gives the data parameters in two domains.

Table 5.13 Information of two datasets

Source Data		Target Data	
1	$\boldsymbol{\mu}_1^s = [1\ 1]$ $\quad \boldsymbol{\sigma}_1^s = \begin{pmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{pmatrix}$	1	$\boldsymbol{\mu}_1^t = [1\ 4]$ $\quad \boldsymbol{\sigma}_1^t = \begin{pmatrix} 0.4^2 & 0.1 \\ 0.1 & 0.4^2 \end{pmatrix}$
2	$\boldsymbol{\mu}_2^s = [1\ 3]$ $\quad \boldsymbol{\sigma}_2^s = \begin{pmatrix} 0.4^2 & 0 \\ 0 & 0.4^2 \end{pmatrix}$	2	$\boldsymbol{\mu}_2^t = [2\ 1]$ $\quad \boldsymbol{\sigma}_2^t = \begin{pmatrix} 0.5^2 & 0.1 \\ 0.1 & 0.5^2 \end{pmatrix}$
3	$\boldsymbol{\mu}_3^s = [3\ 2]$ $\quad \boldsymbol{\sigma}_3^s = \begin{pmatrix} 0.3^2 & 0 \\ 0 & 0.3^2 \end{pmatrix}$	3	$\boldsymbol{\mu}_3^t = [3\ 3]$ $\quad \boldsymbol{\sigma}_3^t = \begin{pmatrix} 0.4^2 & 0.1 \\ 0.1 & 0.4^2 \end{pmatrix}$

Table 5.14 Accuracy of models in predicting the labels in domains

Accuracy of model M^s in source domain	98.60%
Accuracy of model M^s in target domain	37.20%
Accuracy of model $\overline{M}^t$ in target domain	78.07%

The experiment results are listed in Table 5.14. Comparing Table 5.11 with Table 5.14, since the variances of datasets also change, the gap between source domain and target domain becomes large, and the accuracy of the existing model in predicting labels of target data in case 3 (37.20%) is smaller than that in case 2 (38.20%). And the accuracy of the new model $\overline{M}^t$ for target domain is 78.07%, much higher than 37.20%.

Experiment B: The pattern of classes in two domains change.

In this part, we consider the situation that the distributions of source data and target data are the same but the pattern of classes has changed, it means for example, half data with label 1 in source domain got label 1 in target domain, half data with label 1 and half data with label 2 in source domain got label 2 in target domain, and half data with label 2 and all data with label 3 in source domain got label 3 in target domain. The partition of datasets in target domain is clearly shown in Figs. 5.13 and 5.14, and in order to compare source and target domain, the partition of source datasets is also shown in Figs. 5.13 and 5.14.

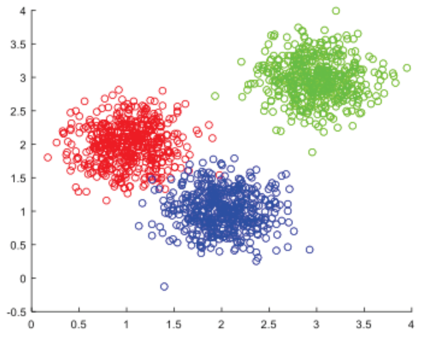


Figure 5.13 Instances in source domain

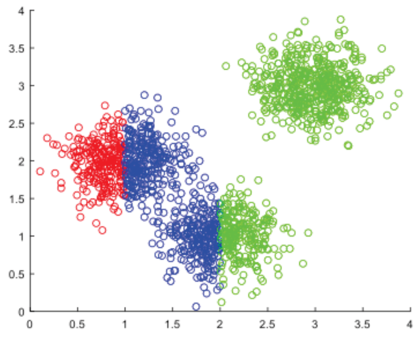


Figure 5.14 Instances in target domain

From Figs. 5.13 and 5.14, we can clearly see the partition of data, where red points represent instances with label 1, blue points represent instances with label 2, and green points represent instances with label 3. Obviously, the partition of data in source domain and target domain are quite different. From the experiment results shown in Table 5.15, we draw the conclusion that our method could deal with the situation that the pattern of classes changes in transfer learning very well.

Table 5.15 Accuracy of models in predicting the labels in domains

Accuracy of model M^s in source domain	99.60%
Accuracy of model M^s in target domain	66.47%
Accuracy of model $\overline{M}^t$ in target domain	92.71%

5.7 Summary

This chapter focuses on the special fuzzy domain adaptation case where the number of fuzzy rules is not identical in the source and target domain. Based on the fact that the performance of transfer learning is good if the number of fuzzy rules in the source domain is no less than that in the target domain, a method is proposed to deal with the situation, where the fuzzy rules do not match in two domains. Actually, this can be created as a criterion when selecting the source domain, i.e. choosing the source domain that has more fuzzy rules than the target domain. Unfortunately, if there are only source domains with less fuzzy rules, we could adopt the approach of reconstructing the source model with more fuzzy rules, and using them to perform the transfer learning procedure. The experiments validate the proposed method when handling the fuzzy rule mismatch case, and analyze the sensitivity of some parameters of constructing the nonlinear mappings for input and output spaces. except for the nonlinear mappings are used to change the input space of the model, piecewise linear mappings are also adopted to modify the existing model. Moreover, the method is used to solve the domain adaptation classification problem with multiple labels, and the results demonstrate a good capability of our method in this scenario.

Chapter 6

Fuzzy Transfer Learning Based on IGMM and Active Learning

6.1 Introduction

Although some fuzzy rule-based methods are presented to cope with the domain adaptation problems, two main issues are still outstanding. One critical factor that affects the performance of the constructed fuzzy models is to correctly determine the number of fuzzy rules to use. Given a set of data points without any information about the data's distribution, it is difficult to find an appropriate number of clusters to differentiate the data. The traditional brute-force approach of trying every number is time-consuming and rather inefficient. The second issue is how to acquire enough labeled data of sufficiently high quality to build a model for the target domain. For example, even if some labeled data is available, if all or most of that data only pertain to one aspect of the domain, the constructed model will contain inherent bias. Hence, in this chapter, we propose a method for dealing with these two issues to improve the accuracy of target models.

The rest of this chapter is organized as follows. Section 6.2 presents the preliminaries of this chapter, including the IGMM and active learning. Section 6.3 details the fuzzy rule-based method and how an IGMM and active learning are used to improve the performance of the target model. Sections 6.4 and 6.5 present the validation tests using synthetic and real-world datasets. Finally, the summary of this chapter is discussed in Section 6.6.

6.2 Preliminaries

This section gives an overview of the IGMM and active learning, which will be applied in the presented method.

6.2.1 The infinite Gaussian mixture model

The finite Gaussian mixture model with k components (k Gaussian distributions) is written as

$$p(y|\mu_1,\mu_2,\ldots,\mu_k,s_1,s_2,\ldots,s_k,\pi_1,\pi_2,\ldots,\pi_k)=\sum_{j=1}^{k}\pi_j N(\mu_j,s_j^{-1}) \qquad (6.1)$$

where μ_j are the means, s_j are the precisions (inverse variances), and π_j are the mixing proportions (which must be positive and sum to one). N is a (normalized) Gaussian distribution with a specified mean and variance, and $y=\{y_1,\ldots,y_n\}$ are the observations. We wish to find the best solution (π_j,μ_j,s_j) with respect to y. However, k needs to be selected by a user and is sometimes sensitive to the training process. Thus, researchers find selecting k automatically a more desirable approach, which means that we need to find the best k even when $k\to\infty$. (Rasmussen, 2000) proposed an IGMM to explore the properties of 6.1 when $k\to\infty$. If we assume μ_j has Gaussian priors $p(\mu_j|\lambda,r)\sim N(\lambda,r^{-1},s_j)$ has

Gamma priors $p(s_j|\beta,\omega) \sim G(\beta,\omega^{-1})$, and π_j is given a symmetric Dirichlet prior (also known as multivariate beta) with a concentration parameter α/k, the limitation of the conditional posterior, when $k \to \infty$, is calculated by

$$\begin{cases} p(c_i = j|c_{-i},\mu_j,s_j,\alpha) \propto \frac{n_{-i,j}}{n-1+\alpha} s_j^{\frac{1}{2}} e^{-\frac{s_j(y_i-\mu_j)^2}{2}}, \text{for component j where } n_{-i,j} > 0 \\ p(c_i \neq c_{i'} \text{for all} i' \neq i|\mathbf{c}_{-i},\lambda,r,\beta,\omega,\alpha) \propto \\ \frac{\alpha}{n-1+\alpha} \int p(y_i|\mu_j,s_j)p(\mu_j,s_j|\lambda,r,\beta,\omega)d\mu_j ds_j \end{cases} \tag{6.2}$$

where c_i is a stochastic indicator variable taking its values from 1 . . . k, $c_{-i} = (c_1,\ldots,c_{i-1},c_{i+1},\ldots,c_n)$, and $n_{-i,j}$ is the number of observations, excluding y_i, that are associated with component j. Thus, we have conditional posterior for a single indicator given all the other indicatorsμ_j and s_j. Using 6.2 and the Gibbs sampling method, we can determine the value of k (i.e., by finding a new class or removing an existing class) based on the posterior probability for a single indicator, which means k can be selected automatically in a one-time sampling process (Rasmussen, 2000). After completing the sampling process T times, the k with the highest frequency is chosen as the final selection.

6.2.2 Active learning

Active learning is a subfield of machine learning. The key hypothesis behind active learning is that the performance of a learning algorithm can be boosted if it is allowed to choose the data from which it learns (Settles, 2010). In supervised machine learning systems, a large number of labeled instances are required to build a model. Sometimes these labels come at little or no cost, but in other cases, obtaining labels can be very difficult, time-consuming, or expensive. Active learning is well-suited to such scenarios, where labeled data is hard to obtain. A variety of different active learning strategies have been used to select unlabeled data for a human annotator to label. However, all these strategies require the "informativeness" of the unlabeled instances to be evaluated through a query strategy, such as uncertainty sampling (Lewis and Catlett, 1994), query-by-committee (Burbidge et al., 2007), expected model change (Settles et al., 2008), expected error reduction (Guo and Greiner, 2007), variance reduction (Settles and Craven, 2008), or a density-weight method (Xu et al., 2007).

6.3 Domain adaptation through active learning

This section presents the framework of our method and the motivation behind each procedure in overview and then in more detail. A theoretical analysis of the method's performance index is also included.

6.3.1 The framework

The proposed method consists of four main procedures: using an IGMM to reveal the structure of the data; applying active learning techniques to augment the information in the target domain; training the model in the source domain; and, finally, executing knowledge transfer in the form of fuzzy rules from the source domain to the target domain. The framework is shown in Fig. 6.1.

When constructing a Takagi-Sugeno fuzzy model, the number of clusters (fuzzy rules) should be known beforehand. However, it is often hard to determine the optimal number of clusters for a specific dataset without additional information. The most recent methods traverse all the numbers in a range and select the one with the best performance – a brute-force approach. However, this approach is costly and, sometimes, finding the optimal range is not easy.

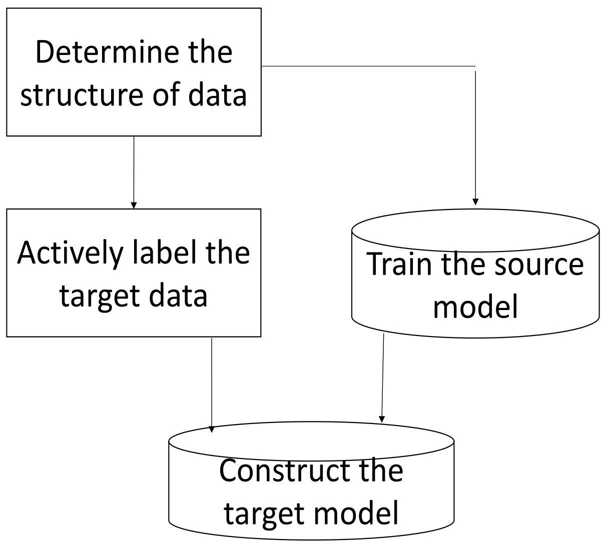

Determine the structure of data
Actively label the target data
Train the source model
Construct the target model

Figure 6.1 The framework of the active learning-based fuzzy transfer learning method

IGMM provides a solution for clustering the data with no necessary prior knowledge to limit the number of clusters to traverse. The idea behind IGMM is to fit the data distribution by mixing Gaussian distributions – a process that can be treated as a data structure detection procedure, which is of benefit to all cluster-based systems.

IGMM's role in the first procedure is to describe the structure of the data, which is beneficial for determining the number of fuzzy rules to use when constructing the subsequent models for both domains. Knowing the data structures in each domain is very important, as it greatly influences the effectiveness of the knowledge transfer process. For example, if the data structures in both domains are very similar, i.e., they have the same number of clusters and the means of the combined Gaussian distributions are close, then we can assume that the source and target domains have a strong corresponding relationship. Conversely, if the source and target domains have different numbers of clusters and the means of the distributions are quite far, it is reasonable to assume they have a weak relationship. To extract useful information, different transfer strategies should be adopted for different domain relationships. Thus, IGMM also provides a basis for guiding the transfer procedure.

After the correlation between two domains has been evaluated, the information in the target domain is assessed to determine the potential knowledge that can be transferred. The data in the target domain are divided into two groups: the instances with labels and the instances without labels. Compared to unlabeled data, labeled data contain more information and have a greater influence on the outcomes of prediction problems, especially regression tasks. Unlike classification tasks, where the results largely depend on the distribution and structure of the data, regression prediction tasks rely on more complicated factors. For instance, in the Takagi-Sugeno fuzzy regression model, the data distributions only determine the conditions of the fuzzy rules, i.e., whether or not each instance adheres to a particular fuzzy rule. The conclusions and the linear functions are governed by other factors that have a more critical impact on the final output. This is also the main reason that unsupervised domain adaptation is infeasible for regression tasks where only unlabeled data are available.

Therefore, labeled target data is necessary for the learning process in domain adaptation problems, and the quality of the labeled data greatly determines the quality of the transfer learning results. Thus, evaluating the quality of the labeled target data is an important part of the process if the results are to be used as a basis for subsequent procedures. Based on the characteristics of Takagi-Sugeno fuzzy model, the aim is to try and ensure the labeled data is a member of as many clusters as possible. If the labeled data in the target domain are spread among all clusters, we can assume the information is sufficient to train a well-performing model. Otherwise, if all the labeled data fall into one cluster or do not overlap with all the clusters, the information in the target domain is considered to be of low quality and needs to be augmented. Employing the IGMM and an active learning technique makes augmenting the information in target domain a reality.

The IGMM's "detection procedure" reveals the cluster structure of the unlabeled data in the target domain. Additionally, the IGMM evaluates the quality of the existing labeled target data, if lacking, the active learning technique adds information by selecting some instances from the unlabeled group for annotation by an expert. The process of selecting the unlabeled target data obeys one of the core principles of active learning: the instances that contain the most information are chosen. The informativeness of each instance is defined, which then supports the following data selection process. Thus, with the help of active learning, the number of the labeled target data is increased, and the information in the target domain is expanded.

The first two procedures can be thought of as pre-processing and preparation steps for constructing the models for each domain in the following step. A Takagi-Sugeno fuzzy model for the source domain is trained, and a set of fuzzy rules is generated. However, due to the discrepancies between the source and target domains, the fuzzy rules in the source domain cannot be directly used with the target data; they need to be modified.

Two approaches are used to modify the existing fuzzy rules: changing the input variables and changing the output variables. Each input variable is assumed to be determined by some hidden features, so the different distributions of input variables in the two domains are due to the different hidden features or different weights of these features. The idea behind changing the input variables is to adjust the number and weights of the hidden features so that the changed input distribution is more compatible with the target data. The approach in changing the output space is to revise the results of the linear functions to make the new rules more suitable for regression tasks in the target domain. These modifications are made through an optimization process.

6.3.2 A transfer learning method based on IGMM and active learning

Consider two domains: a source domain with a large amount of labeled data, and a target domain with very little labeled data. The dataset in the source domain is denoted as $\boldsymbol{D} = \{(\boldsymbol{x}_1^s,\ y_1^s), \cdots, (\boldsymbol{x}_{N_s}^s,\ y_{N_s}^s)\}$, where $\boldsymbol{x}_k^s \in R^n$ $(k = 1, \cdots, N_s)$ is an n-dimensional input variable, the label $y_k^s \in R$ is a continuous variable, and N_s indicates the number of data pairs. The dataset in the target domain $\boldsymbol{H}$ consists of two subsets: one with labels and one without. $\boldsymbol{H} = \{\boldsymbol{H}_L, \boldsymbol{H}_U\} = \{\{(\boldsymbol{x}_1^t, y_1^t), \cdots, (\boldsymbol{x}_{N_{t1}}^t, y_{N_{t1}}^t)\},\ \{\boldsymbol{x}_{N_{t1+1}}^t, \cdots, \boldsymbol{x}_{N_t}^t\}\}$, where $\boldsymbol{x}_k^t \in R^n$ $(k = 1, \cdots, N_t)$ is the n-dimensional input variable, $y_k^t \in R$ is the label only accessible for the first N_{t1} data. $\boldsymbol{H}_L$ includes the instances with labels, and $\boldsymbol{H}_U$ contains the data without labels. The number of instances in $\boldsymbol{H}_L$ and $\boldsymbol{H}_U$ are N_{t1} and $N_t - N_{t1}$ respectively, and satisfy $N_{t1} \ll N_t$, $N_{t1} \ll N_s$.

In this problem setting, a well-performing model can be built for the source domain because there are sufficient labeled data. However, that model cannot be used directly to solve regression tasks in the target domain because the rules needs to be modified to fit the target data first. The following steps outline a fuzzy rule-based transfer learning method, based on IGMM and active learning, to modify the source model for use with the target data.

Step 1. Applying IGMM to discover the structure of data

This procedure reveals the data structure of both domains, with two benefits. First, identifying the data structures of each domain provides insights into the relationships between the data, which can be used to guide the transfer learning procedure. Second, understanding the data structures is conducive to selecting the most informative labeled data for the target domain.

Through two separate parses of an unsupervised learning process, IGMM simulates the distributions of the data in the source and target domains. The input data are $\{\boldsymbol{x}_1^s, \cdots, \boldsymbol{x}_{N_s}^s\}$ and $\{\boldsymbol{x}_1^t, \cdots, \boldsymbol{x}_{N_t}^t\}$. IGMM's exploration process is illustrated in the following example.

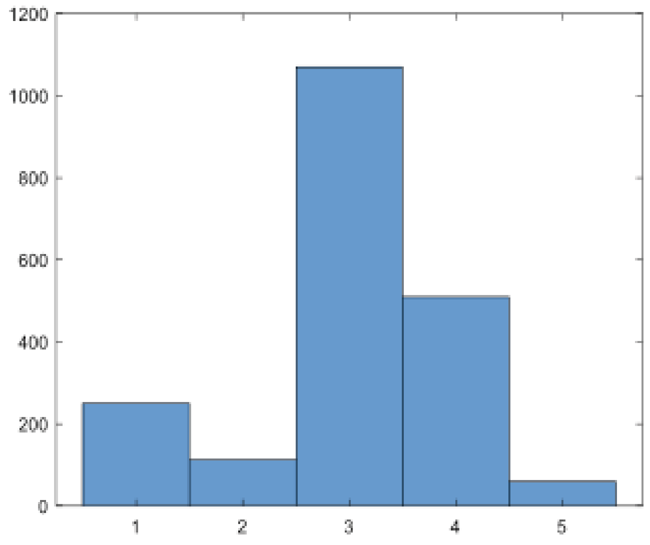


Figure 6.2 Example of the results for IGMM

Fig. 6.2 shows the probabilities of various data structures in a dataset in histogram form. The x-coordinate represents the number of Gaussian distributions, i.e., the number of clusters, and the y-coordinate represents the number of times that the dataset is divided into the corresponding clusters. As the figure shows, in 2000 iterations of IGMM, the dataset was divided into three clusters more than 1000 times, into four clusters about 500 times, into one cluster about 250 times, and into two or five clusters less than 100 times. Therefore, we can conclude, with high probability, that the dataset is composed of three Gaussian distributions (clusters).

Suppose the cluster range for the source domain is [csmin, csmax], where csP has the highest probability. Similarly, the range for the target domain is [ctmin, ctmax], with ctP as the highest probability. Comparing the histograms of the data reveals insights into the relationship between the two domains that can be used to select an appropriate transfer strategy. For instance, if csP is equal to ctP, then the corresponding parameters of the Gaussian distributions are also similar, which means the source and target domains are close, and the knowledge of source domain is likely to be highly beneficial in constructing the model for the target domain. Additionally, the number csP (=ctP) provides a good guide as to the number of fuzzy rules to use to build the prediction models for each domain. Further, this type of analysis can also be used to inform domain selection. For example, given multiple source domains to choose from, comparing the data structure of each candidate may reveal the most suitable match for the target domain. And, if the none of the data structures match, i.e., csP is not identical to ctP, a different transfer strategy can be explored. However, even with the results of the structural analysis, determining the optimal number of fuzzy rules to use when constructing the models may still be difficult. Therefore, a brute-force approach across a reduced range of cluster numbers may be required to select the one that delivers the best performance. The range for the number of clusters is:

$$[cmin, cmax] \tag{6.3}$$

where $cmin = \max\{2, \min\{csmin, ctmin\}\}$, and $cmax = \min\{csmax, ctmax\}$. Restricting $cmin$ to not less than 2 ensures the nonlinearity of the model.

Step 2. Using active learning to augment the labeled target data

The purpose of this procedure is to increase the amount information in the target domain by actively selecting and labeling some of the data.

The procedure begins with an evaluation of the existing labeled target data. Given the analysis in Step 1, suppose we choose to construct the model with three fuzzy rules $c(=3)$. FCM clusters the input data for the target domain into the membership matrix U. The index of H_L in Hdetermines the membership of all the labeled target data to all the clusters. The membership matrix for the labeled target data is denoted as U_L:

$$U_L = \begin{pmatrix} 0.2 & 0.7 & 0.1 \\ 0.7 & 0.1 & 0.2 \\ 0.5 & 0.2 & 0.3 \end{pmatrix} \tag{6.4}$$

The number of labeled data in each cluster is counted, with each instance counted in the cluster with the highest membership. The statistic result is as follows:

$$SU_L = \begin{pmatrix} 2 & 1 & 0 \end{pmatrix} \tag{6.5}$$

The first two clusters contain labeled data, but the third cluster does not, so active learning is used to augment the information in the target domain and, hopefully, populate this cluster.

"Informativeness" is the key criteria for which data to select for labeling in each cluster. Essentially, informativeness is a measure of information contained in the data, and the level of informativeness is highly dependent on the cluster it is grouped with, i.e., an instance will have a different level of informativeness in different clusters. A concrete instance $\boldsymbol{x}_k^t$ in the ith cluster is highly informative if $\boldsymbol{x}_k^t$'s membership to the ith cluster is high. Thus, the informativeness of $\boldsymbol{x}_k^t$ in cluster i is defined as

$$I_i(\boldsymbol{x}_k^t) = U_{ki} \tag{6.6}$$

Further, a threshold d determines the minimum number of labeled target data needed for each cluster. Unlabeled data assessed as being highly informative according to the above definition are then selected and sent to experts to be annotated. Taking the example above, the $d-2$ unlabeled data with the highest informativeness in the first cluster are selected for labeling. Similarly, $d-1$ and d unlabeled data in the remaining two clusters are selected for labeling.

At the end of Step 2, the number of labeled target data has increased from N_{t1} to $3d$.

Step 3. Constructing a prediction model for the source domain

This procedure governs the construction of the source model M^s based on the source dataset $\boldsymbol{D}$. The formulation for model M^s follows
model M^s

$$\text{if } \boldsymbol{x}^s \text{ is } A_i(\boldsymbol{x}^s,\ \boldsymbol{v}_i^s),\ \text{then } y^s \text{ is } L_i(\boldsymbol{x}^s, \boldsymbol{a}_i^s) \qquad i = 1, 2, \ldots, c \tag{6.7}$$

where $\boldsymbol{v}_i^s$ are the centers of the clusters that determine the conditions of the fuzzy rules, and $\boldsymbol{a}_i^s$ are the coefficients of the linear functions of the input variables that govern the conclusions of the fuzzy rules.

The number of fuzzy rules c depends on the results of the analysis in Step 1. If the data structure of the (chosen) source domain is similar to the data structure of the target domain, c is set to $csP(=ctP)$. If the data structures are divergent, c is taken from the range $[cmin, cmax]$.

Given a sufficient amount of labeled source data, a well-performing prediction model M^s for the source domain can be built. However, the target domain will invariably contain different data distributions to the source data, and the model M^s would perform poorly if trained on the target data without prior modifications, which leads to Step 4.

Step 4. Modifying the existing fuzzy rules to fit the target data

Through this procedure, the hidden features in model M^s are adjusted, including the amounts and weights, to modify the input space so that the distributions of the input variables are more compatible with target data. This approach is based on the assumption that the input variables in both domains have similar, or even the same, hidden features.

The mappings with network structure are still applied to modify the input space. The neurons in the hidden layers represent the connotative features to be used as input variables. The transformation of these neurons through the layers modifies their weights to ultimately construct input variables with new meanings and distributions that can be used to build a new target model $\overline{M}^t$.

After the mapping procedure, the fuzzy rules are transformed into

$$\text{if } \boldsymbol{x}_k^t \text{ is } A_i(\Phi(\boldsymbol{x}_k^t),\ \boldsymbol{v}_i^s),\ \text{then } y_k^s \text{ is } L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s) \qquad i = 1, 2, \cdots, c \tag{6.8}$$

With this modified input space, the new fuzzy rules, including the cluster centers and the linear coefficients can be used to predict outputs with the target data.

Φ is mapped through an optimization procedure using the target data, which was augmented in Step 2 to ensure a sufficient level of labeled data to construct a well-performing model. Therefore, once optimized, the cost function in 6.9 minimizes the distance between the outputs of model $\overline{M}^t$ and the real values in the target data. The cost function follows:

$$S = \sqrt{\frac{1}{N_{t1}} \sum_{k=1}^{N_{t1}} \left(\sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}_k^t), \Phi(\boldsymbol{v}_i^s)) L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s) - y_k^t \right)^2} + \frac{\lambda_2}{2} w^T w \tag{6.9}$$

The overall algorithm for the proposed method described above is provided in Algorithm 2.

Algorithm 2. Domain adaptation procedure based on active learning

```
Input: $\boldsymbol{D}$, $\boldsymbol{H}$, Output: $\boldsymbol{Y}_U$ for $\boldsymbol{H}_U$ 1. Apply IGMM to $\{\boldsymbol{x}_1^s, \cdots, \boldsymbol{x}_{N_s}^s\}$ and $\{\boldsymbol{x}_1^t, \cdots, \boldsymbol{x}_{N_t}^t\}$
2. Decide the number of clusters $c$
3. Apply active learning to augment target information
    3.1 give the threshold $d$
    3.2 validate the current labeled target data
    3.3 find out the data with most information in each cluster
    3.4 select the target data and label them
4. Train source model $M^s$
5. Modifying the existing model to get $M^t$
6. Use $M^t$ to predict the response $\boldsymbol{Y}_U$ for $\boldsymbol{H}_U$
```

6.3.3 Performance index

This section provides the formulations for the performance indexes of the constructed models used to evaluate the proposed method.

Each model is constructed through five-fold cross-validation. The source model M^s is trained on the dataset $\boldsymbol{D}$, and the root mean square error (RMSE) of M^s on training set is calculated by

$$Q = \sqrt{\frac{1}{N_{s1}} \sum_{k=1}^{N_{s1}} (\sum_{i=1}^{c} A_i(\boldsymbol{x}_k^s, \boldsymbol{v}_i^s) L_i(\boldsymbol{x}_k^s, \boldsymbol{a}_i^s) - y_k^s)^2} \tag{6.10}$$

where $\boldsymbol{v}_i^s$ are the centers of the clusters and $\boldsymbol{a}_i^s$ are the linear functions of the fuzzy rules . $\boldsymbol{x}_k^s$ is the input variable for the source data, and $\sum_{i=1}^{c} A_i(\boldsymbol{x}_k^s, \boldsymbol{v}_i^s) L_i(\boldsymbol{x}_k^s, \boldsymbol{a}_i^s)$ is the corresponding output of the model M^s. y_k^s is the real output for $\boldsymbol{x}_k^s$.

The probability of model M^s is tested on the target data with

$$Q1 = \sqrt{\frac{1}{N_t} \sum_{k=1}^{N_t} (\sum_{i=1}^{c} A_i(\boldsymbol{x}_k^t, \boldsymbol{v}_i^s) L_i(\boldsymbol{x}_k^t, \boldsymbol{a}_i^s) - y_k^t)^2} \tag{6.11}$$

where $\boldsymbol{v}_i^s$ and $\boldsymbol{a}_i^s$ are the parameters of source model. $\boldsymbol{x}_k^t$ is the input variable of target data, and $\sum_{i=1}^{c} A_i(\boldsymbol{x}_k^t, \boldsymbol{v}_i^s) L_i(\boldsymbol{x}_k^t, \boldsymbol{a}_i^s)$ is the corresponding output of the model M^s. y_k^t is the real output for $\boldsymbol{x}_k^t$.

The target model $\overline{M}^t$ is tested on the unlabeled dataset to verify the generalizability of the constructed model with

$$Q2 = \sqrt{\frac{1}{N_t - d} \sum_{k=1}^{N_t - d} (\sum_{i=1}^{c} A_i(\Phi(\boldsymbol{x}_k^t), \Phi(\boldsymbol{v}_i^s) L_i(\Phi(\boldsymbol{x}_k^t), \boldsymbol{a}_i^s) - y_k^t)^2} \tag{6.12}$$

where $\boldsymbol{x}_k^t \in H_U$ are the data without labels. The real labels y_k^t are only available in the testing procedure.

6.4 Experiments on synthetic datasets

In this section, we present the experiments conducted with synthetic datasets to validate the proposed method. We specifically evaluated the impact of the IGMM and the active learning technique on the performance of the constructed models. Each experiment was conducted in two stages. The first stage tested the IGMM's ability to explore the structure of data as a basis for the knowledge transfer procedure. The second stage tested the ability of the active learning technique to optimally augment information in the target domain.

6.4.1 Exploring data's structure using IGMM

The design of a well-performing transfer learning algorithm depends on the relationship between the source and target data. In this experiment, we considered two cases to simulate different relationships between the domains. The first case assumes that the source and target data stem from identical domains and, therefore, their distributions are quite similar. The second case considers two quite different source and target domains, where there is a great divergence in the data structures between each domain. A different transfer strategy was applied in each case, and IGMM plays a different role in each scenario.

6.4.1.1 Similar source and target data

Here, the target data was derived from the source domain, so the distributions and structures of data in both domains are very similar. Thus, IGMM's role is to provide guidance with domain selection – a challenging issue in transfer learning. For instance, suppose there are multiple source domains, but only one domain is the most suitable for the target domain. If IGMM is able to easily find the most suitable source domain, it would effectively improve the transfer learning performance.

Three groups of experiments were executed to illustrate the role of IGMM in selecting a source domain. In each group of experiments, the data in the target domain was generated with a different number of clusters, and three source domains with various numbers of clusters were prepared.

In the first experiment, IGMM was used to analyze the structure of the data in these domains. From Fig. 6.3, we can see that the three source domains are divided into two, three, and four clusters, respectively, with the highest probabilities. Moreover, the target data consists of two clusters with the highest probability.

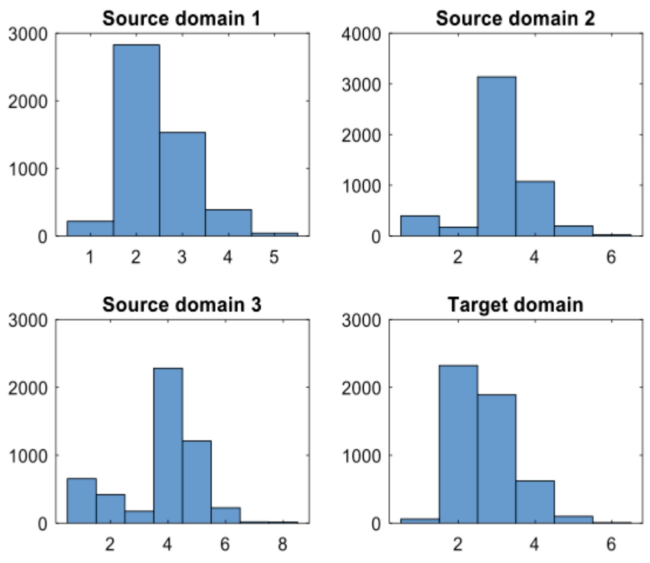


Figure 6.3 The data structure in Experiment 1

In analyzing the results of IGMM, the first source domain has the same number of clusters as the target domain and should be the best choice for the target domain. To verify this conclusion, knowledge from the three source domains was transferred, in turn, to the target domain, and we compared the results to assess the transfer performance. These results are shown in Table 6.1.

Table 6.1 Various source domains and a two-clusters target domain

datasets		Q	Q1	Q2
Source 1 (2)	Target (2)	0.0441± 0.0006	2.2774± 0.0000	1.8487± 0.0350
Source 2 (3)	Target (2)	0.0250± 0.0002	2.2743± 0.0000	1.8549± 0.0397
Source 3 (4)	Target (2)	0.1023± 0.0009	2.5293± 0.0003	1.9929± 0.0481

The first two columns in Table 6.1 represent the datasets for the source and target domains. The number in the brackets indicates the number of clusters in that dataset. All models were constructed through five-fold cross-validation; therefore, the results in the last four columns are displayed in the form of “mean$\pm$variance”. The third column is the RMSE of the source model on the source data, and a low error means a well-performing regression prediction model was produced for the source domain. The fourth column is the RMSE of source model on the target data, which indicates that the source model is not compatible with target data. The results in column five show the performance of the target models constructed using the proposed methods: changing the input. The mean values in the fifth column are less than in the fourth column, which means that the proposed method has greatly improved the prediction accuracy of the existing model in the target domain. In particular, comparing the results in the fifth column, we can conclude that the first source domain showed the best performance and has the same number of clusters as the target domain. This finding validates the role of IGMM in selecting the most suitable domain in cases where the source and target domain have very similar structures.

A further experiment was designed to assess the application of IGMM. The target data was generated with three clusters, and three source domains were generated with two, three and four clusters. The data structures for the three source domains and the target domain are shown in Fig. 6.4, and the performance of each of the resulting IGMMs is displayed in Table 6.2.

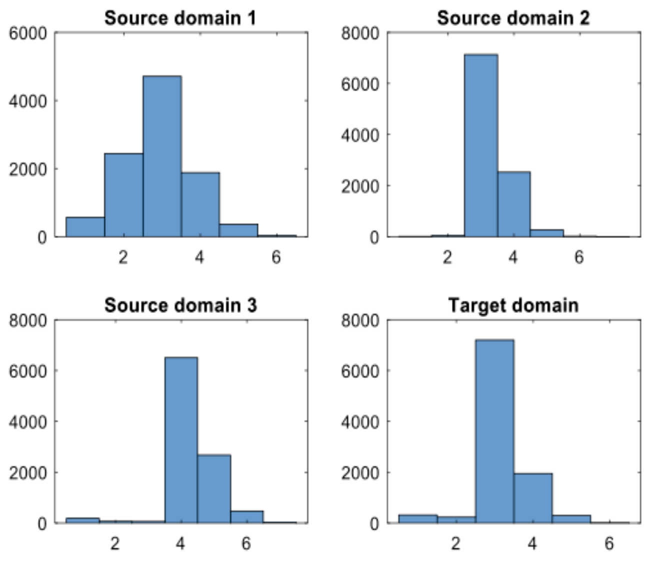


Figure 6.4 The data structure in Experiment 2

Table 6.2 Various source domains and a three-clusters target domain

datasets		Q	Q1	Q2
Source 1 (2)	Target (3)	0.0301± 0.0005	1.0320± 0.0001	0.8560± 0.0108
Source 2 (3)	Target (3)	0.0196± 0.0000	0.4586± 0.0000	0.7997± 0.0482
Source 3 (4)	Target (3)	0.0505± 0.0086	2.0170± 0.0067	0.8678± 0.0024

The next experiment included four datasets: three source domains, each with a different number of clusters and one target domain. The data structures resulted from the IGMM are shown in Fig. 6.5, and the performance of the constructed models appears in Table 6.3.

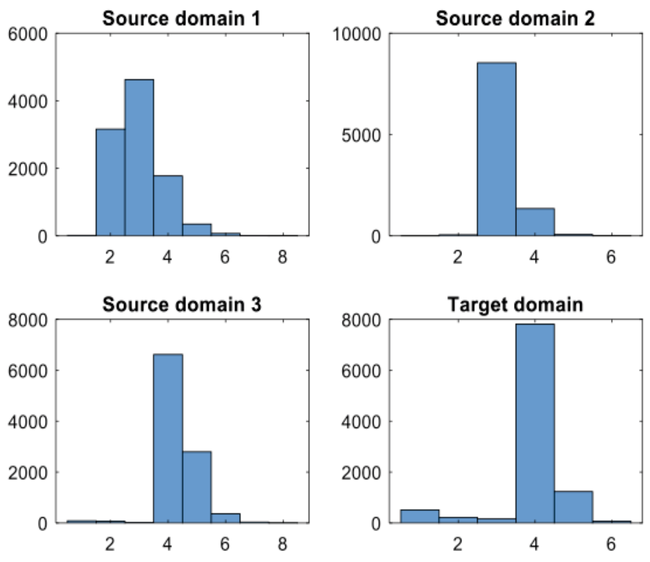


Figure 6.5 The data structure in Experiment 3

Table 6.3 Various source domains and a four-clusters target domain

datasets		Q	Q1	Q2
Source 1 (2)	Target (4)	0.0153± 0.0001	3.0610± 0.0001	1.6237± 0.0043
Source 2 (3)	Target (4)	0.0930± 0.0384	3.6952± 0.0586	1.4924± 0.0415
Source 3 (4)	Target (4)	0.1104± 0.0142	1.9950± 0.0069	1.1537± 0.0066

We reached the same conclusion from the second and third experiments: that the results from the IGMM both improve the performance of the constructed target model and provide useful clues for the domain selection process. This conclusion, therefore, validates the role of IGMM in selecting a suitable domain in cases where the source and target domain have very similar structures.

The above three experiments tell us that the transfer learning has an obvious effect when the number of clusters (fuzzy rules) in the source and target domains is identical. However, the distance of the data can also affect transfer learning. Hence, the next experiment was designed to explore the impact of data distance on the model's performance.

The target dataset was generated first, and the source datasets were generated based on the target data by gradually increasing the gap between the centers of the clusters and the linear functions. The center of the clusters and the linear functions in both domains follow the relation below:

$$\boldsymbol{v}_i^s = \boldsymbol{v}_i^t(1+\varepsilon)\boldsymbol{a}_i^s = \boldsymbol{a}_i^t(1+\varepsilon) \tag{6.13}$$

where ε is an constant that controls the increment, and ε is a crucial parameter that controls the degree of difference between the source and target data.

Table 6.4 Various source domains and a four-clusters target domain

ε	Q1	Q2
0.05	0.2851 ± 0.0000	0.2299±0.0055
0.1	0.5788 ± 0.0000	0.1931±0.0037
0.15	0.8833 ± 0.0000	0.2168±0.0024
0.2	1.1956 ± 0.0000	0.2602±0.0007
0.25	1.5246 ± 0.0000	0.1903±0.0002
0.3	1.8561 ± 0.0000	0.2517±0.0013
0.35	2.1902 ± 0.0000	0.2108±0.0004
0.4	2.5384 ± 0.0000	0.2607±0.0057
0.45	2.8882 ± 0.0000	0.3365±0.0173
0.5	3.2610 ± 0.0000	0.3403±0.0003

As ε increases, the divergence between the source and target data becomes greater. The value of ε was set to increase from 0.05 to 0.5 in steps of 0.05. Thus, ten source datasets were generated and knowledge was transferred from each source dataset to the same target dataset. Here, we only used the method of changing the input space as an illustration. The results are shown in Table 6.4 and

Fig. 6.6 clearly displays the changing tendency in model performance when as the value of ε changes.

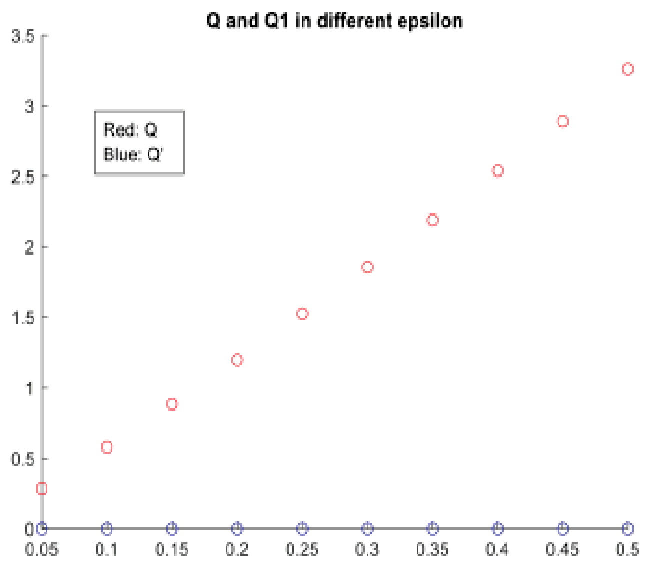


Figure 6.6 The model's performance with varying values for ε

The red circles indicate the RMSE of the source model on the target data. The increasing trend shows that with an increase of ε, the discrepancy between the source data and target data becomes greater, and the source model becomes more mismatched to target data. The blue circles represent the performance of the target model using the proposed method. There are no obvious changes, but effectiveness of the proposed method is still verified. These results also indicate good transfer learning performance if the data structures of the two domains are similar.

6.4.1.2 Different source and target

With this set of experiments, we evaluated IGMM's performance in cases where the source and target domains have very different data structures, i.e., where each domain has a different number of clusters (fuzzy rules). Three separate domain adaptation experiments were conducted. Each time, one of the datasets was selected as the target domain, and the remaining two datasets were treated as the source domains. We used the approach of traversing all the clusters in source and target domains to determine the optimal number of clusters.

The three datasets with different colors in Fig. 6.7 are considered. Fig. 6.7 shows that each of the three datasets has a different number of clusters and a different data structure. The analytical results from IGMM for the three datasets are shown in Fig. 6.8. The results of these experiments are shown in Tables 6.5 - 6.6.

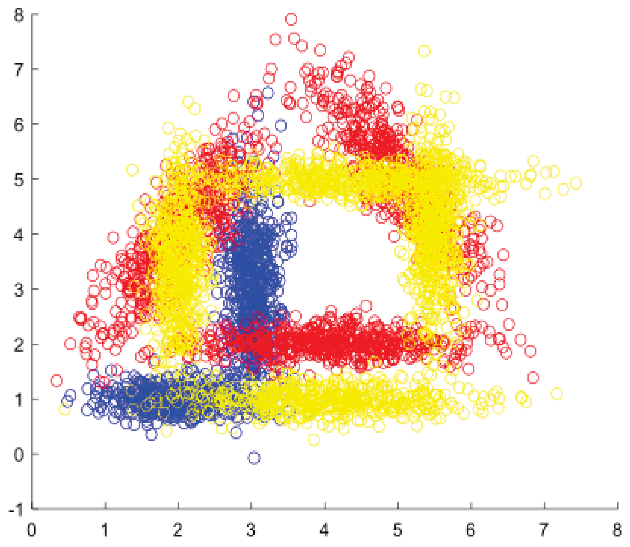


Figure 6.7 Three datasets with different numbers of clusters

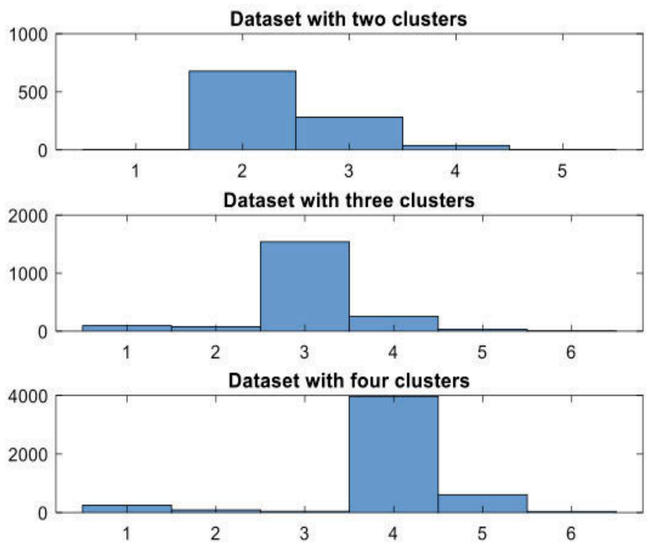


Figure 6.8 The structure of the three datasets

Table 6.5 The values of Q2 with two clusters target domain

clusters	4r to 2r	3r to 2r
2	1.5003±0.4920	0.6256±0.0031
3	2.0345±0.2321	0.6150±0.1162
4	**0.5992±0.0089**	0.8732±0.5013
5	0.7783±0.0353	**0.4578±0.0245**
6	0.9645±0.1183	0.8543±0.3426

Table 6.6 The values of Q2 with three clusters target domain

clusters	2r to 3r	4r to 3r
2	1.5431±0.0213	3.0206±0.0375
3	1.5438±0.0210	2.8064±0.4057
4	**1.4310±0.0172**	4.7779±0.2630
5	1.4653±0.0024	**1.7450±0.0041**
6	1.5910±0.0024	2.3647±0.0507

Table 6.7 The values of Q2 with four clusters target domain

clusters	2r to 4r	3r to 4r
2	**1.0372±0.0072**	1.4666±0.0163
3	1.6847±0.0079	1.4228±0.0025
4	1.2896±0.0143	1.3803±0.0123
5	1.6691±0.0064	**1.3791±0.0077**
6	1.5846±0.0237	1.4657±0.0768

The results in Tables 6.5 - 6.6 do not reveal an obvious rule for determining the optimal number of clusters in the domain adaptation process. Thus, in cases where the source and target domains have very different structures, the brute-force approach of trying all the numbers and selecting the one with the best performance remains the best option.

6.4.2 Augmenting the information in the target domain with an active learning technique

The three experiments in this section were designed to verify the use of active learning technique in improving the performance of the built target model. In each experiment, the source and target datasets were generated with the same number of fuzzy rules (two, three, and four respectively), and all the labeled target data were selected from one cluster. The experimental results are shown in Table 6.8.

Table 6.8 Exploring the effect of the active learning technique

Clusters	Q1	Q2 (no active learning)	Q2 (active learning)
2	1.3676± 0.0001	1.4596± 0.0202	1.0731± 0.0075
3	0.3534± 0.0001	0.9250± 0.0320	0.8760± 0.0157
4	2.0330± 0.0026	2.1865± 0.0653	1.7536± 0.1304

Comparing values of the third and fourth columns in the three experiments, we found that using the active learning technique significantly enhances the accuracy of the target model constructed using the proposed method.

In addition, we conducted the above experiments with different values of d to determine the impact of d on the performance of the presented method.

Table 6.9 The values of Q2 with varying clusters

d	Datasets (two clusters)	Datasets (three clusters)	Datasets (three clusters)
5	1.0731±0.0075	0.8760±0.0157	1.7536±0.1304
10	1.0768±0.0124	0.8518±0.0180	1.9097±0.0295
15	1.0076±0.0023	0.9004±0.0380	0.6976±0.0034
20	0.9702±0.0117	0.9019±0.0055	1.3242±0.0072

The results in Table 6.9 show that the performance of the constructed target model does not display an increasing trend as the value of d increases, which indicates that d does not play a critical role in the active learning-based domain adaptation method.

Since IGMM identifies the data structure of the dataset, i.e., the number of clusters, few instances could represent the information of one cluster, and the design of our algorithm satisfies the requirement of covering the information in all clusters. Thus, the results in Table 6.9 are reasonable and acceptable. Further, these results are a promising signal for good transfer learning performance with little labeled target data.

6.5 Experiment on real-world datasets

Studies of regression problems in domain adaptation are scarce, so there are no public datasets available to verify the proposed method. Hence, we used real-world datasets from the UCI Machine Learning Repository and modified them to simulate various regression domain adaptation problems. A detailed description of these modifications follows.

The first dataset concerns "air quality". We selected two of the existing attributes, "temperature" and "relative humidity", as the input data and chose "absolute humidity" as the output. All the attributes were normalized, and the dataset was split into two domains based on "relative humidity". Data with a "relative humidity" of greater than 0.5 were chosen as the source domain, and the remaining data were used to form the target domain. Further, the two attributes in the source data were all perturbed by random numbers following a normal distribution $N(0.1, 0.1)$, and the two attributes in the target data were perturbed by the normal random numbers following $N(7, 1)$ and $N(5, 1)$, respectively. There were 3600 labeled instances in the source domain and 1200 instances in the target data; 10 were labeled.

Although a target domain may only contain a small amount of labeled data, it can still be used to train a model. However, we assert that a model trained solely on a small amount of labeled data will not perform well. And, to support this assertion, we trained a target model with various levels of labeled target data and tested its performance denoted by "QT".

We used the IGMM to identify the data structures in the “air quality” dataset and show the results in Fig. 6.9.

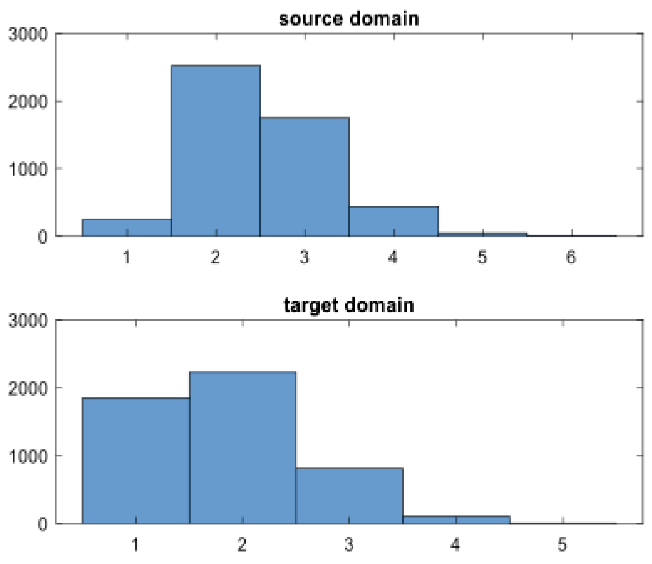


Figure 6.9 IGMM’s results with the "air quality" dataset

From Fig. 6.9, we can see that the data in the source domain and target domain are both divided into two clusters with the greatest probability, so two is the optimal number of fuzzy rules to construct models and implement transfer learning. To verify this conclusion, we executed the proposed domain adaptation method with varying numbers of fuzzy rules, and compared the results, as shown in Table 6.10.

Table 6.10 The values of Q2 with varying clusters for "air quality"

Clusters	Q1	Q2	QT	Q3
2	0.1241±0.0000	0.2575±0.0000	0.4153±0.0060	0.1075±0.0000
3	0.1237±0.0000	0.2568±0.0000	0.3217±0.0204	0.1097±0.0000
4	0.1235±0.0000	0.2618±0.0000	0.1970±0.0034	0.1076±0.0000
5	0.1232±0.0000	0.2616±0.0000	0.2319±0.0046	0.1085±0.0000

In all experiments, the value for Q3 was smaller than for Q2 and QT, which indicates that the model built using our method is superior to both the existing source model and the model built using only labeled target data. Additionally, the small variances indicate that the models built using the proposed method have good generalizability. Comparing the values of Q3 with different clusters, we find that the transfer learning method has the best performance with two fuzzy rules. In addition, the number of labeled target data increased with an increase in the number of clusters due to the active learning technique. These results show that determining the appropriate number of fuzzy rules is significantly more important than accumulating more labeled data.

We conducted the same experiment on the "housing dataset", which aims to predict the "MEDV" (the median value of owner-occupied homes in US$1000's) using six input attributes. The data was normalized and split into two datasets using the attribute "TAX", which represents the full-value property tax rate per $10,000. Instances of "TAX" smaller than 0.5 were used to form the source dataset, and instances of "TAX" larger than 0.5 were used as the target dataset. The attributes "RM", "AGE", and "B" of the source data were perturbed by random numbers taken from $N(0.1, 0.1)$, while the same attributes in the target data were perturbed by normal random numbers using the distributions $N(7, 1)$, $N(5, 1)$ and $N(8, 1)$, respectively. There were 360 labeled instances in the source domain and 130 instances in the target data; 10 were labeled.

Again, IGMM was used to identify the data structure. The results are shown in Fig. 11.

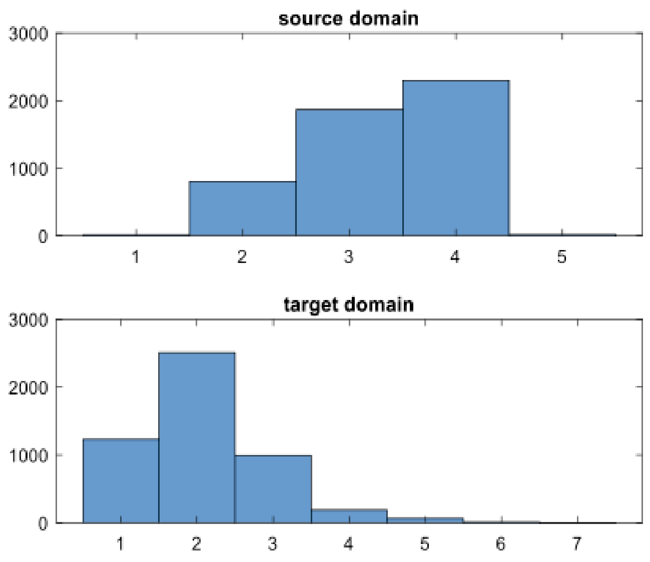


Figure 6.10 IGMM's results with the "housing" dataset

Unlike the first dataset, where it was easy to determine the number of fuzzy rules, in this dataset, the probability distributions of the clusters in the source and target domains are quite different. Although the source data and target data were derived from the same domain, our modifications resulted in quite different data distributions in each domain. Based on our analysis of IGMM's results, we decided to try all the numbers of clusters as a cross-check of IGMM's performance. The results are shown in Table 6.11.

Table 6.11 The values of Q2 with varying clusters for "housing"

Clusters	Q1	Q2	T	Q3
2	0.1098±0.0003	0.2558±0.0003	0.6306±0.1175	0.1799±0.0002
3	0.1006±0.0001	0.2280±0.0005	0.2931±0.0179	0.1713±0.0002
4	0.0913±0.0002	0.1801±0.0003	0.3670±0.0093	0.2276±0.0000
5	0.0902±0.0001	0.1844±0.0002	0.2297±0.0040	0.1827±0.0002
6	0.1044±0.0017	0.2504±0.0074	0.2053±0.0007	0.2325±0.0003
7	0.0920±0.0001	0.3463±0.0266	0.2850±0.0088	0.1813±0.0001

From Table 6.11, we can see that the constructed target model performed best with three fuzzy rules. There was no obvious trend with a change in the number of fuzzy rules. Therefore, traversing all the numbers of clusters and choosing the best one remains the best method for choosing the optimal number of fuzzy rules in cases where the source and target data greatly diverge.

6.6 Summary

This chapter presents a method of discovering the structure of data and actively augmenting information in a target domain to improve the performance of fuzzy rule-based domain adaptation. IGMM is used to explore the relationship between the data structures in the source and target domains and provide guidance on a domain selection and transfer strategy. The idea of active learning is applied to increase the amount of labeled information in target domain by actively labeling the most informative data in the source domain for use in the target domain. A set of experiments on synthetic datasets verifies both the positive effect of IGMM and the active learning technique on the transfer learning process. Additionally, promising results on real-world datasets validate the effectiveness of the proposed domain adaptation method in practical settings.

Chapter 7

Fuzzy Rule-based Domain Adaptation in Heterogeneous Spaces

7.1 Introduction

Different with the homogeneous domain adaptation problems, the heterogeneous domain adaptation problems are more complex and challegeable due to the different feature spaces in two domains. Therefore, extracting a common feature space shared between domains is always a necessary procedure prior to knowledge transfer. According to the amount of the labeled data in the target domain, the heterogeneous domain adaptation problems can be divided into two types: semi-supervised domain adaptation and unsupervised domain adaptation. Most of the existing work is dedicated to the semi-supervised type, and only few literatures focus on unsupervised domain adaptation in heterogeneous space, which is a very difficult issue.

The heterogeneous semi-supervised domain adaptation models have been widely researched in recent years. But, heterogeneous unsupervised domain adaptation models are rarely studied due to their current limitations: the feature spaces are heterogeneous, and no labeled data in the target domain could provide information. Kernel canonical correlation analysis (KCCA) was proposed to address these problems when there are paired instances in the source and target domains, but this method is not valid when there are no paired instance in both domain.

All the methods for the heterogeneous domain adaptation concentrate on the classification problems, and none of them is proposed to handle the regression prediction problems. This chapter focus on using the fuzzy rule-based model for the heterogeneous domain adaptation for regression tasks. As we described in Chapter 1, regression problem is much more difficult and challengeable than the classification problem, since the outputs of regression model is depended on not only the input variables, but also some hidden factors. Therefore, we need some labeled target data to induce the specific knowledge of target domain, and a semi-supervised domain adaptation method in heterogeneous space using fuzzy models is presented in this chapter.

The rest of this chapter is organized as follows. Section 7.2 formalizes the problem we aim the solve. Section 7.3 presents the details of the heterogeneous domain adaptation method and the corresponding algorithm. The experiments on Section 7.4 validate the effectiveness of the presented fuzzy rule-based method in dealing with the regression domain adaptation in heterogeneous space, and analyze the sensitivity of some parameters of the proposed model. Moreover, this method is also used to solve some real-world problems. Finally, the summary of this chapter is discussed in Section 7.5.

7.2 Problem statement

The dataset in the source domain is denoted by $\boldsymbol{D} = \{(\boldsymbol{x}_1^s, y_1^s), \cdots, (\boldsymbol{x}_{N_s}^s, y_{N_s}^s)\}$, where $\boldsymbol{x}_k^s \in R^{n_s}$, $k = 1, \cdots, N_s$, is the n_s-dimensional input variable, the label $y_k^s \in R$ is the continuous output variable, and N_s indicates the number of data. Since the amount of source data with labels is massive, a well-performing regression model for the source domain – the TS fuzzy model M^s – can be learned.
model M^s

$$\text{if } \boldsymbol{x}^s \text{ is } A_i(\boldsymbol{x}^s,\ \boldsymbol{v}_i^s),\ \text{then } y^s \text{ is } L_i(\boldsymbol{x}^s, \boldsymbol{a}_i^s) \qquad i = 1, 2, ..., c_s \tag{7.1}$$

The dataset in the target domain contains two subsets: one with labels and one without labels $\boldsymbol{H} = \{\boldsymbol{H}_L, \boldsymbol{H}_U\} = \{\{(\boldsymbol{x}_1^t, y_1^t), \cdots, (\boldsymbol{x}_{N_{t1}}^t, y_{N_{t1}}^t)\}, \{\boldsymbol{x}_{N_{t1+1}}^t, \cdots, \boldsymbol{x}_{N_t}^t\}\}$, where $\boldsymbol{x}_k^t \in R^{n_t}$, $k = 1, \cdots, N_t$ is the n_t-dimensional input variable, $y_k^t \in R$, $k = 1, \cdots, N_{t1}$ is the continuous output variable. $\boldsymbol{H}_L$ includes the instances with labels, and $\boldsymbol{H}_U$ contains the data without labels. The numbers of data in $\boldsymbol{H}_L$ and $\boldsymbol{H}_U$ are N_{t1} and $N_t - N_{t1}$ respectively, and satisfy $N_{t1} << N_t$, $N_{t1} << N_s$.

Suppose the ideal model for the target domain is M^t.
model M^t

$$\text{if } \boldsymbol{x}^t \text{ is } A_i(\boldsymbol{x}^t,\ \boldsymbol{v}_i^t),\ \text{then } y^t \text{ is } L_i(\boldsymbol{x}^t, \boldsymbol{a}_i^t) \qquad i = 1, 2, ..., c_t \tag{7.2}$$

Building a well-performing TS fuzzy model needs a large amount of data with labels, and inadequately data in $\boldsymbol{H}_L$ cannot guarantee performance of the constructed model in the target domain. Furthermore, discrepancies between the source and target data mean that using the source model to solve target tasks is impossible.

Clarifying the divergence between the source and target data plays a crucial role in conducting knowledge transfers from the source domain to the target domain. Since the proposed methods aim to adapt the domains using fuzzy rule-based models, we differentiate the source domain and target domain using the feature space and fuzzy rules. In general, the difference between the source and target domains is summarized to the four cases shown in Fig. 7.1.

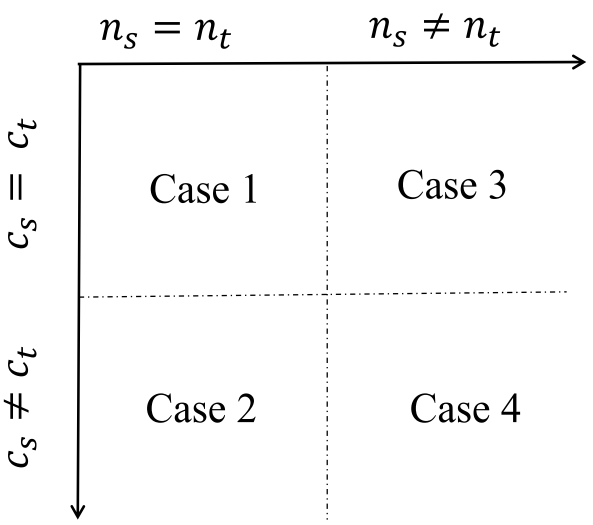


Figure 7.1 Four cases distinguishing the source and target domains

(1) Case 1: $n_s = n_t$, and $c_s = c_t$. The input spaces (feature spaces) in the source and target domains have the same dimensionality with different distributions, and the number of constructed fuzzy rules is also equal in both domains.

(2) Case 2: $n_s = n_t$, and $c_s \neq c_t$. The input spaces in both domains have the same dimension with different distributions, but the number of fuzzy rules is different.

(3) Case 3: $n_s \neq n_t$, and $c_s = c_t$. Discrepancies in the input data in both domains occur in dimensionality and in distribution. However, there is an equal number of fuzzy rules in both domains.

(4) Case 4: $n_s \neq n_t$ and $c_s \neq c_t$. This is the most complicated case. The input space, in dimensionality and in distribution, and the number of fuzzy rules are both different in the source and target domains.

Based on the definition of domain adaptation, Case 1 and Case 2 belong to homogeneous domain adaptation, and Case 3 and Case 4 fall into the scope of heterogeneous domain adaptation. Chapters 3 and 4 solve the domain adaptation problems in Case 1, Chapter 5 handles Case 2, and Chapter 6 copes with Cases 3 and 4.

7.3 Fuzzy domain adaptation in heterogeneous spaces

The greatest challenge in heterogeneous domain adaptation is the different dimensions of the input spaces in both domains. This means that the distributions of the input variables are not only different but also the number of the input variables. To eliminate the gap caused by the mismatch of the feature spaces, many studies in heterogeneous domain adaptation employ a method that extracts a latent feature space that is shared between both domains. This space can then be used to facilitate knowledge exploration and transfer between the domains. After projecting the input data of the two domains into the latent feature space, the heterogeneous domain adaptation problem is converted to a homogeneous domain adaptation problem. In our method, we also use an extraction approach.

As discussed in Section 6.2, Cases 3 and 4 both belong to the category of heterogeneous domain adaptation. The distinction between them is the number of the fuzzy rules in the two domains. In Case 3, the number of fuzzy rules is equal; in Case 4, it is not. After transforming the data from the original feature space to the latent feature space, some information will be lost, and we cannot guarantee the number of fuzzy rules will remain unchanged in the latent feature space. However, the relation is not limited by the number of fuzzy rules in the domains. As long as there are a sufficient number of constructed fuzzy rules in the latent feature space of the source domain, they can be modified and used for tasks in the target domain.

Therefore, the proposed FHeDA method for solving Case 3 and Case 4 follows with no discrimination. The FHeDA method contains three steps for transferring knowledge from the source domain to the target domain.

Step 1: Extract the latent feature space and map all the input data to it.

Since the primary factor impeding knowledge transfer across the domains is a mismatched feature space, the first step is to map the source and target data into a uniform feature space, where common features can benefit from the discovery and transfer of the knowledge. This minimizes the gap between the distributions of the input variables for both domains. Approximating the input data distributions across both domains has two benefits:

(a) The conditions of the fuzzy rules are dominated by the center of the clusters, which are derived from the input data using a clustering algorithm. In turn, the relative location of the center of the clusters greatly influences the construction of the fuzzy rules. Converting the input data into a common latent feature space forces the data distribution in each domain to approximate the other, which reduces the difference between the relative location of the center of the clusters in both domains.

(b) Projecting to the latent feature space also facilitates knowledge transfer of the fuzzy rule conclusions, which are represented as the linear functions of input variables. Similar distributions restrict the input variables in approximate ranges and reduce disagreement in the input data in both domains.

In this chapter, the **c**anonical **c**orrelation **a**nalysis (CCA) algorithm Hardoon et al. (2004) has been used to derive the latent feature space. The latent feature space is extracted by learning a mapping between the original feature space and the latent feature space. Based on the input data $\{\boldsymbol{x}_1^s, \cdots, \boldsymbol{x}_{N_s}^s\}$ and $\{\boldsymbol{x}_1^t, \cdots, \boldsymbol{x}_{N_t}^t\}$, two mappings U_s and U_t are learned simultaneously to convert the input data from the original feature spaces of two domains to the latent feature space.

Under the mappings U_s and U_t, the input data in two domains will have a new representation as follows:

$$U_s(\boldsymbol{x}_k^s) = \overline{\boldsymbol{x}}_k^s,\ k = 1, \cdots, N_s \tag{7.3}$$

$$U_t(\boldsymbol{x}_k^t) = \overline{\boldsymbol{x}}_k^t,\ k = 1, \cdots, N_t \tag{7.4}$$

Therefore, the input data $\{\boldsymbol{x}_1^s, \cdots, \boldsymbol{x}_{N_s}^s\}$ and $\{\boldsymbol{x}_1^t, \cdots, \boldsymbol{x}_{N_t}^t\}$ in two domains becomes $\{\overline{\boldsymbol{x}}_1^s, \cdots, \overline{\boldsymbol{x}}_{N_s}^s\}$ and $\{\overline{\boldsymbol{x}}_1^t, \cdots, \overline{\boldsymbol{x}}_{N_t}^t\}$ in the latent feature space, and the dimension of the latent feature space is $\overline{n} = \min(n_s, n_t)$.

Step 2: Build a TS fuzzy model for the source domain in the latent feature space.

The dataset $\boldsymbol{D}$ in the source domain has become $\overline{\boldsymbol{D}} = \{(\overline{\boldsymbol{x}}_1^s, y_1^s), \cdots, (\overline{\boldsymbol{x}}_{N_s}^s, y_{N_s}^s)\}$, and a TS fuzzy model $\overline{M}^s$ is built for the source domain in the latent feature space.

model $\overline{M}^s$

$$\text{if } \overline{\boldsymbol{x}}_k^s \text{ is } A_i(\overline{\boldsymbol{x}}_k^s,\ \overline{\boldsymbol{v}}_i^s),\ \text{then } y_k^s \text{ is } L_i(\overline{\boldsymbol{x}}_k^s, \overline{\boldsymbol{a}}_i^s) \qquad i = 1, 2, \cdots, \overline{c} \tag{7.5}$$

where $\overline{c}$ is the critical parameter that determines the amount of fuzzy rules in the source domain and in the target domain. This parameter is so pivotal, Section 6.4 presents a set of experiments designed to explore the impact of $\overline{c}$ on the model's construction.

Although the discrepancy between the input data's distributions in the two domains has been reduced in the latent feature space, a gap still exists and cannot be completely eliminated. Moreover, different linear functions (the conclusions of the fuzzy rules) are another factor that distinguish the source and target domains, so model $\overline{M}^s$ in the source domain cannot be directly used to solve regression tasks in the target domain.

Step 3: Modify the existing fuzzy rules in model $\overline{M}^s$ to make them suitable for the target data.

In the latent feature space, it is difficult to detect which parts of the two domains' fuzzy rules are different. Using the same strategy as in homogeneous domain adaptation, we modify the source model in three different ways and choose the one with the best performance on the target data.

model $\overline{M}_1^t$

$$\text{if } \overline{\boldsymbol{x}}_k^t \text{ is } A_i(\Phi(\overline{\boldsymbol{x}}_k^t),\ \overline{\boldsymbol{v}}_i^s),\ \text{then } y_k^s \text{ is } L_i(\Phi(\overline{\boldsymbol{x}}_k^t), \overline{\boldsymbol{a}}_i^s) \qquad i = 1, 2, \cdots, \overline{c} \tag{7.6}$$

model $\overline{M}_2^t$

$$\text{if } \overline{\boldsymbol{x}}_k^t \text{ is } A_i(\overline{\boldsymbol{x}}^t),\ \overline{\boldsymbol{v}}_i^s,\ \text{then } y_k^s \text{ is } \Psi_i(L_i(\overline{\boldsymbol{x}}_k^t, \overline{\boldsymbol{a}}_i^s)) \qquad i = 1, 2, \cdots, \overline{c} \tag{7.7}$$

model $\overline{M}_3^t$

$$\text{if } \overline{\boldsymbol{x}}_k^t \text{ is } A_i(\Phi(\overline{\boldsymbol{x}}^t),\ \overline{\boldsymbol{v}}_i^s),\ \text{then } y_k^s \text{ is } \Psi_i(L_i(\Phi(\overline{\boldsymbol{x}}_k^t), \overline{\boldsymbol{a}}_i^s)) \qquad i = 1, 2, \cdots, \overline{c} \tag{7.8}$$

where $\Phi = [\Phi_1 \cdots \Psi_{\overline{n}}]$, and $\Phi = [\Psi_1 \cdots \Psi_{\overline{c}}]$ are the transformation mappings for the input space and output space.

The final target model $\overline{M}^t$ is chosen from the best among the models $\overline{M}_1^t$, $\overline{M}_2^t$ and $\overline{M}_3^t$, i.e.,

$$\overline{M}^t = \overline{M}^t_i, \text{if}\, \overline{M}^t_i \geq \overline{M}^t_j,\ i,j = 1.2.3 \tag{7.9}$$

where $\overline{M}^t_i \geq \overline{M}^t_j$ means the performance of $\overline{M}^t_i$ on the target data $\boldsymbol{H}_U$ is no worse than $\overline{M}^t_j$.

The construction of the mappings for the input and output spaces is the same as for homogeneous domain adaptation, i.e., using a network to modify each input or output variable. The parameters of the mappings are derived by minimizing the following cost functions.

When changing the input space to get model $\overline{M}^t_1$, the cost function below is minimized

$$W1 = \sqrt{\frac{1}{N_{t1}}\sum_{k=1}^{N_{t1}}\Big(\sum_{i=1}^{\bar{c}}\frac{A_i(\Phi(\overline{\boldsymbol{x}}^t_k),\Phi(\overline{\boldsymbol{v}}^s_i))}{\sum_{j=1}^{\bar{c}}A_j(\Phi(\overline{\boldsymbol{x}}^t_k),\Phi(\overline{\boldsymbol{v}}^s_j))}L_i(\Phi(\overline{\boldsymbol{x}}^t_k),\overline{\boldsymbol{a}}^s_i) - y^t_k\Big)^2} + \frac{\lambda_2}{2}w^T w \tag{7.10}$$

The cost function includes two terms: one aims to decrease the gap between the output of the constructed target model and the data's real labels, and the other is a structural risk term to control the complexity of the built model. Here, only the target data with labels are applied to modify the existing model. The reason target data without labels is not used is that the transformation from the original feature space to the latent feature space may change the manifold of the input space. The data that is close in distance in the latent feature space may be far from each other in the original feature space. So using neighboring target data without labels in the latent feature space to improve the result is risky, and may bring negative impact to the performance of the constructed model. This is checked experimentally in Section 6.4.

When changing the output space, the cost function $W2$ is minimized

$$W2 = \sqrt{\frac{1}{N_{t1}} \sum_{k=1}^{N_{t1}} \Big(\sum_{i=1}^{\bar{c}} \frac{A_i(\overline{\boldsymbol{x}}_k^t, \overline{\boldsymbol{v}}_i^s)}{\sum_{j=1}^{\bar{c}} A_j(\overline{\boldsymbol{x}}_k^t, \overline{\boldsymbol{v}}_j^s)} \Psi_i(L_i(\overline{\boldsymbol{x}}_k^t, \overline{\boldsymbol{a}}_i^s)) - y_k^t \Big)^2} + \frac{\lambda_2}{2} w^T w \qquad (7.11)$$

Similarly, when changing the input and output space simultaneously, the cost function $W3$ is minimized

$$W3 = \sqrt{\frac{1}{N_{t1}} \sum_{k=1}^{N_{t1}} \Big(\sum_{i=1}^{\bar{c}} \frac{A_i(\Phi(\overline{\boldsymbol{x}}_k^t), \Phi(\overline{\boldsymbol{v}}_i^s))}{\sum_{j=1}^{\bar{c}} A_j(\Phi(\overline{\boldsymbol{x}}_k^t), \Phi(\overline{\boldsymbol{v}}_j^s))} \psi_i(L_i(\Phi(\overline{\boldsymbol{x}}_k^t), \overline{\boldsymbol{a}}_i^s)) - y_k^t \Big)^2} + \frac{\lambda_2}{2} w^T w \qquad (7.12)$$

The overall algorithm corresponding to the FHeDA method is provided in Algorithm 3.

Algorithm 3. Heterogeneous domain adaptation procedure

```
Input: $\boldsymbol{D}$, $\boldsymbol{H}$,
Output: $\boldsymbol{Y}_U$ for $\boldsymbol{H}_U$
1. Use CCA to learn $U_s$ and $U_t$
2. Map $\boldsymbol{x}_k^s$ to $\overline{\boldsymbol{x}}_k^s$, and map $\boldsymbol{x}_k^t$ to $\overline{\boldsymbol{x}}_k^t$
3. Train model $\overline{M}^s$ using $\overline{\boldsymbol{D}}$
4. Modify the fuzzy rules in $\overline{M}^s$
    4.1 Change input space to get $\overline{M}_1^t$
    4.2 Change output space to get $\overline{M}_2^t$
    4.3 Change both input and output spaces to get $\overline{M}_3^t$
5. Compare $\overline{M}_1^t$, $\overline{M}_2^t$, and $\overline{M}_3^t$, and choose the best one as $\overline{M}^t$
6. Use $\overline{M}^t$ to predict the output $\boldsymbol{Y}_U$ for $\boldsymbol{H}_U$
```

7.4 Empirical results analysis

Both synthetic and real-world datasets are used to validate the FHeDA method for the heterogeneous domain adaptation problems.

7.4.1 Experiments on synthetic datasets

This section comprises three subsections. Information about the synthetic datasets and the experimental settings are provided, followed by the experimental results. The second subsection validates the effectiveness of the FHeDA method, and the third subsection discusses the impact of a critical parameter on the model's construction.

7.4.1.1 Datasets and experimental settings

Four datasets of different dimensions were generated according to different numbers of fuzzy rules to simulate various source and target domains in heterogeneous spaces. Information about the generated datasets is provided in Table 7.1. For example, 'dataset 2' contains 3-dimensional data, which was generated according to 4 fuzzy rules.

Table 7.1 Information of four datasets

	dimension	number of fuzzy rules
dataset 1	3	3
dataset 2	3	4
dataset 3	4	3
dataset 4	4	4

The datasets in Table 7.1 were assembled to simulate six cases for experimentation in heterogeneous domain adaptation as outlined in Table 7.2. The second and third columns indicate the origin of the source data and target data. The fourth column shows the dimensionality of the source data and target data, while the last columns show the number of fuzzy rules present in each domain.

Table 7.2 Heterogeneous domain adaptation datasets

	Source domain	Target domain	dim(S) vs dim(T)	rules(S) vs rules (T)
Exp 1	dataset 1	dataset 3	3 vs 4	3 vs 3
Exp 2	dataset 1	dataset 4		3 vs 4
Exp 3	dataset 2	dataset 3		4 vs 3
Exp 4	dataset 3	dataset 1	4 vs 3	3 vs 3
Exp 5	dataset 3	dataset 2		3 vs 4
Exp 6	dataset 4	dataset 1		4 vs 3

7.4.1.2 Regression results

The RMSE of the models on the target data $\boldsymbol{H}_U$ for the six experiments are shown in Table 7.3.

From the results in Table 7.3, we can conclude that the models $\overline{M}_1^t$, $\overline{M}_2^t$, and $\overline{M}_3^t$ are all superior to the existing source model $\overline{M}^s$. In the six experiments, model $\overline{M}_2^t$ showed the best performance in half the experiments, with model $\overline{M}_3^t$ performing the best in the other half. Although model $\overline{M}_1^t$ did not surpass the other two models, we intend to retain it as an alternative model. The three models show vast differences in performance on different datasets, and the availability of different options for modifying an existing model enhance the probability of our method to successfully fit the data to the target domain.

Table 7.3 Results of the six experiments

Source to target	RMSE of the models				
	$\overline{M}^s$	$\overline{M}_1^t$	$\overline{M}_2^t$	$\overline{M}_3^t$	$\overline{M}^t$
Exp 1	9.5862± 0.0002	3.3168± 0.0021	2.9422± 0.0359	3.1510± 0.2120	2.9422± 0.0359
Exp 2	9.8785± 0.0003	7.0799± 0.0375	5.0758± 0.4040	4.7611± 0.0445	4.7611± 0.0445
Exp 3	8.2935± 0.0008	3.9622± 0.0333	3.2467± 0.1174	2.6774± 0.0090	2.6774± 0.0090
Exp 4	9.2541± 0.0009	2.9019± 0.3178	2.9033± 0.2476	2.5138± 1.0901	2.5138± 1.0901
Exp 5	9.6183± 0.0002	8.8253± 18.5659	3.9799± 1.8011	4.6508± 1.8893	3.9799± 1.8011
Exp 6	9.6353± 0.0001	2.6874± 0.1060	2.5636± 0.0141	2.5900± 0.0040	2.5636± 0.0141

Further, to verify the claim that using target data without labels is risky, we conducted comparative experiments and provide those results in Table 7.4. The results in columns with “W” represent models that were constructed with the help of target data $\boldsymbol{H}_U$. These results indicate that using target data without labels has a negative function in model construction, especially in the methods that change the output space and change both the input and output space. Lower values are shown in bold.

Table 7.4 Using/not using $\boldsymbol{H}_U$ - comparative experiments

	RMSE of the models					
	$\overline{M}_1^t$	$\overline{M}_1^t(W)$	$\overline{M}_2^t$	$\overline{M}_2^t(W)$	$\overline{M}_3^t$	$\overline{M}_3^t(W)$
Exp 1	3.3168± 0.0021	**3.3153**± **0.0020**	2.9422± 0.0359	2.9487± 0.0380	3.1510± 0.2120	**3.1020**± **0.1001**
Exp 2	7.0799± 0.0375	7.0860± 0.0405	5.0758± 0.4040	5.1412± 0.4184	4.7611± 0.0445	4.9762± 0.0084
Exp 3	3.9622± 0.0333	**3.9518**± 0.0121	3.2467± 0.1174	**3.2265**± 0.1688	2.6774± 0.0090	2.7944± 0.0505
Exp 4	2.9019± 0.3178	3.2590± 0.5608	2.9033± 0.2476	3.0016± 0.2099	2.5138± 1.0901	2.9480± 0.5821
Exp 5	8.8253± 18.5659	**8.6197**± 16.5161	3.9799± 1.8011	**3.4271**± 0.3989	4.6508± 1.8893	**4.1819**± 0.1494
Exp 6	2.6874± 0.1060	**2.5686**± 0.0532	2.5636± 0.0141	2.5778± 0.0167	2.5900± 0.0040	2.7160± 0.0332

7.4.1.3 Parameter sensitivity analysis

In the FHeDA method, the number of fuzzy rules used to construct the model in the latent feature space for the source domain is a significant parameter, because this also determines the number of fuzzy rules for the target domain. We also conducted the six experiments described in the last subsection with a varying $\overline{c}$. Table 7.5 shows the impact of $\overline{c}$ on model $\overline{M}_1^t$, as an example.

From the results, we can see that model $\overline{M}_1^t$'s performance was slightly different when $\overline{c}$ was assigned with different values. The RMSE variance as $\overline{c}$ changes is small in "Exp 1, 2, 3, and 6". But when $\overline{c}$ is assigned with 5, 7, and 8, the RMSE variances are large in a few cases in "Exp 4" and in almost all cases in "Exp 5". We attribute this to too little target data with labels – so little data with labels, there isn't enough to represent the characteristics of the entire target dataset. The results in Table 7.3 for "Exp 5" also verify this.

Table 7.5 Results of the sensitivity analysis for $\overline{c}$

$\overline{c}$	RMSE of the models					
	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
4	3.3001± 0.0110	6.9353± 0.0141	3.1835± 0.0019	3.1793± 0.1117	5.0963± 1.6004	2.8555± 0.0290
5	3.2807± 0.0079	6.9348± 0.0130	3.1703± 0.0022	3.7702± 3.1078	5.8903± 5.2418	2.7982± 0.0124
6	3.2774± 0.0064	6.9372± 0.0134	3.1649± 0.0193	3.0173± 0.0524	5.3352± 5.4235	2.8711± 0.0438
7	3.2842± 0.0090	6.9370± 0.0129	3.1738± 0.0252	4.2752± 6.1490	4.5622± 0.3267	2.7861± 0.0172
8	3.2806± 0.0084	6.9346± 0.0129	3.2164± 0.0030	3.7475± 3.3048	5.5040± 8.9666	2.8241± 0.0095

7.4.2 Experiments on real-world datasets

Three real-world datasets from the UCI Machine Learning Repository were used to validate the FHeDA method in heterogeneous domain adaptation. Like the experiments in the homogeneous domain adaptation, the datasets from machine learning area are modified to simulate the scenarios of heterogeneous domain adaptation problems. The detailed description of these datasets is given.

The “Concrete Compressive Strength” dataset aims to predict the concrete compressive strength based on eight features, such as cement content, blast furnace slag, fly ash. This dataset is dedicated to general regression tasks, so it needed to be revised in several respects to simulate a heterogeneous domain adaptation problem. First, the dataset was split into a source domain and a target domain using the “age” feature; instances with an age of less than 100 were treated as data in the target domain, the remainder fell into the source domain. To exacerbate the gap between the source and target domains, the features “blast furnace slag”, “fly ash” and “superplasticizer” were perturbed with random numbers following the normal distributions $N(0.1,0.1)$ and $N(5,1)$ for source data and target data, respectively. The feature “age” in the source domain was then removed creating heterogeneous spaces across the two domains. Ultimately, we arrived at two datasets: one 7-dimensional source domain containing 110 data with labels, and one 8-dimensional target domain including 30 with labels and 80 without labels.

In the “Istanbul stock exchange”, two attributes “stock exchange returns” and the “Istanbul stock exchange national 100 index” were used to predict the “MSCI emerging marks index”. The first 200 instances fell into the source domain, and the last 200 instances were used as the target data. The two features were perturbed with random numbers following normal distribution $N(0.1,0.1)$ for the source data and $N(5,1)$ for the target data. Further, the first feature in the source domain was discarded. Again, we arrived at the two datasets: one 1-dimensional source domain containing 200 data with labels, and one 2-dimensional target domain including 30 with labels and 170 data without labels.

In the last dataset, "air quality", the features "temperature" and "relative humidity" were chosen as the input data, and "absolute humidity" was chosen as the output. The dataset was split based on the "relative humidity" value. Data with a "relative humidity" of greater than 0.5 formed the source domain, the remaining data was used for the target domain. The second feature in the target domain was discarded. Both features in the source domain were perturbed with random numbers following the normal distributions $N(0.1,0.1)$ and $N(7,1)$, respectively. The input data of the target domain was changed with random numbers following normal distributions $N(0.1,0.1)$. The final two datasets were: one 2-dimensional source domain containing 1200 data with labels, and one 1-dimensional target domain including 30 with labels and 1170 without labels.

To further prove our assertion that a model will not perform as well when trained only using a small amount of target data with labels in heterogeneous domain adaptation settings, we constructed and compared four models. The first was built using source data in a latent feature space $\overline{M}^s$. The second was constructed using insufficient target data with labels in the original feature space $\tilde{M}_1^t$. The third was built using target data with labels in a latent feature space $\tilde{M}_2^t$. And the fourth was built using the proposed FHeDA method.

From the results presented in Table 7.6, we can see that the performance of models $\tilde{M}_1^t$ and $\tilde{M}_2^t$ on the target data was poor, confirming our assumption that a model does not perform as well when being trained on less data. The target model $\overline{M}^t$ built using our method was superior to both the existing source model, and the model constructed using only a small amount of target data with labels.

Table 7.6 Results for FHeDA on real-world datasets

Datasets	RMSE of the models			
	$\overline{M}^s$	$\overline{M}_1^t$	$\overline{M}_2^t$	$\overline{M}^t$
Concrete compressive strength	74.70± 3968.07	13144.46± 8.00e+08	17.85± 36.33	2.10± 0.86
Istanbul stock exchange	0.17± 0.00	102.90± 7314.52	1352.42± 2.03e+06	0.14± 0.00
Air quality	0.15± 0.00	0.14± 0.00	186.08± 1.73e+05	0.14± 0.00

7.5 Summary

Heterogeneous domain adaptation problem is a significant type in transfer learning because it is more general in practice, so it has attracted much attention of research laboratories, such as reinforcement learning. This chapter has presented a heterogeneous knowledge transfer method based on fuzzy system. A latent feature space is extracted to minimize the gap between the feature spaces of the two domains. Heterogeneous domain adaptation is converted into homogeneous domain adaptation after mapping all the input data from both domains into the new latent feature space. Experiments completed on synthetic and real-world datasets verify that the proposed methods greatly improve the performance of existing models when solving regression tasks in the target domain in the heterogeneous domain adaptation settings.

The presented methods offer three avenues for modifying the existing source model: changing the input space, changing the output space, or changing both. The model with the best performance is subsequently selected as the target model. Future studies will explore an algorithm that can recognize the differences between the fuzzy rules in the two domains in advance, so we can intentionally adopt a specific algorithm to modify an existing model.

Chapter 8

Conclusion and Future Research

This chapter concludes the thesis and provides some further research directions for this topic.

8.1 Conclusions

Transfer learning has attracted much attention in the area of machine learning due to its powerful knowledge transfer ability and the need for less training data when constructing a prediction model - especially for a newly emerging area. Even though transfer learning has gained considerable attention and is undergoing rapid development, its ability to extract abstract knowledge or transfer knowledge to a domain without labeled data, such as transferring knowledge between two quite different domains, and adapting to a totally new area, is still lacking. Yet, knowledge transfer is one of most natural abilities of humans in order to adapt to changing environments. Thus, as a part of artificial intelligence, transfer learning has a great potential to be explored from the theoretical and practical aspects. The findings of this study are summarised as follows:

(a) This work proposes a fuzzy rule-based homogeneous domain adaptation method (Chapter 3) to achieve research objectives 1 and 3. The proposed method modifies the input space of the target data through mappings to ensure the fuzzy rules of the source model are compatible with the regression tasks in the target domain. The construction procedure of the mappings is guided by an optimization process using labeled target data. The performance of this presented method is influenced by the optimization algorithms, and the experiment results on comparing PSO and DE show that DE is superior to PSO in both stability and optimization performance. Additionally, learning mappings for each input variable is better than constructing mappings for input variables together. The results on real-world datasets validate the effectiveness of our method in solving domain adaptation problems in practice.

(b) This work proposes a granular domain adaptation framework (Chapter 4) to achieve research objectives 2 and 3. The presented granular transfer learning framework provides guidance for the different fuzzy domain adaptation cases: where conditions differ, conclusions differ, or both differ between the source and target domains. The methods can handle domain adaptation problems not only for the regression tasks, but also have a good performance when solving the classification problems. Although there is a specific algorithm for each case, the experiments show that every method has the ability to cope with all the cases, and the method of changing the output shows a powerful transfer capability.

(c) This work proposes a novel fuzzy domain adaptation method (Chapter 5) to achieve research objective 3. A fuzzy rule-based method is proposed to cope with cases of mismatch in the fuzzy rules in domain adaptation, i.e. the numbers of fuzzy rules in the two domains are not identical. The performance of the TS fuzzy model with a greater number of fuzzy rules is superior with a high probability than the performance of a model with a fewer number of fuzzy rules.

Thus, this can be treated as a principle for the source domain's selection. The experiment results validate that the proposed method can handle the mismatch of fuzzy rules in domain adaptation well.

(d) This work proposes a combined transfer learning method (Chapter 6) to achieve research objectives 2 and 3. IGMM and active are applied to enhance the performance of fuzzy rule-based domain adaptation. IGMM explores the relationship between the data structures in the source and target domains and provide guidance on a domain selection and transfer strategy. The idea of active learning is applied to increase the amount of labeled information in target domain by actively labeling the most informative data in the target domain.

(e) This work proposes a heterogeneous domain adaptation method (Chapter 7) to achieve research objective 4. CCA is used to extract a latent feature space shared between the source and target domains. In the latent feature space, the distribution gap is minimized and the heterogeneous domain adaptation problem becomes a homogeneous domain adaptation problem. Since the relation of the fuzzy rules of the two domains in the latent feature space is not clear, it is hard to tell in which parts the fuzzy rules in the two domains are differ, the conditions or the conclusion. The proposed method adopts the strategy of trying all the three algorithms and selects the one with the best performance. And the experiments on real-word datasets validate the practicability of the proposed method.

8.2 Future study

Although some work has been done to solve the regression tasks in the transfer learning area, the current methods still lack the ability to transfer knowledge in an unsupervised learning and extract a good latent feature space for both the source and target domains. Additionally, although the proposed methods have been used to solve real-world datasets, they still lack efficiency when dealing with big data. This thesis identifies the following directions as future work:

Extract a latent feature space to maintain the relation of the input variables and the output, which means the relation of the input variables in each fuzzy rule will remain. Meanwhile, the features in the latent features should have the property of being more abstract than the original features in the source and target domains.

The unsupervised domain adaptation problem, where no labeled data are available in the target domain, will be developed. Compared with the classification task, unsupervised domain adaptation in regression tasks is more challenging, since the output depends on not only the input data, but also some hidden factors. Therefore, how to explore the extra information beyond the input data in an unsupervised situation is critical. The relation between the source domain and target domain will be re-explored to extract more information to assist the knowledge transfer for the regression tasks in an unsupervised learning process.

This parallel implementation will be developed to accelerate the transfer learning speed with the help of a parallel machine. Current transfer learning methods will be carefully redesigned and assigned to multi-threads (multi-core parallel machines) or multi-machines (distributed parallel machines). Theoretical support could also be distributed along with the task assignment of the original problem.

Bibliography

Ahmed, A., Yu, K., Xu, W., Gong, Y., and Xing, E. (2008). Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In *European Conference on Computer Vision*, pages 69–82. Springer.

Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., and Spyropoulos, C. D. (2000). An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–167. ACM.

Baralis, E., Chiusano, S., and Garza, P. (2008). A lazy approach to associative classification. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):156–171.

Bargiela, A. and Pedrycz, W. (2016). Granular computing. In *HANDBOOK ON COMPUTATIONAL INTELLIGENCE: Volume 1: Fuzzy Logic, Systems, Artificial Neural Networks, and Learning Systems*, pages 43–66. World Scientific.

Behbood, V., Lu, J., and Zhang, G. (2011). Long term bank failure prediction using fuzzy refinement-based transductive transfer learning. In *2011 IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 2676–2683. IEEE.

Behbood, V., Lu, J., and Zhang, G. (2013a). Fuzzy bridged refinement domain adaptation: Long-term bank failure prediction. *International Journal of Computational Intelligence and Applications*, 12(01):1350003.

Behbood, V., Lu, J., and Zhang, G. (2013b). Text categorization by fuzzy domain adaptation. In *2013 IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 1–7. IEEE.

Behbood, V., Lu, J., and Zhang, G. (2014). Fuzzy refinement domain adaptation for long term prediction in banking ecosystem. *IEEE Transactions on Industrial Informatics*, 10(2):1637–1646.

Behbood, V., Lu, J., Zhang, G., and Pedrycz, W. (2015). Multistep fuzzy bridged refinement domain adaptation algorithm and its application to bank failure prediction. *IEEE Transactions on Fuzzy Systems*, 23(6):1917–1935.

Bellman, R. E. and Zadeh, L. A. (1970). Decision-making in a fuzzy environment. *Management Science*, 17(4):B–141.

Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 17–36.

Berkan, R. C. and Trubatch, S. (1997). *Fuzzy system design principles*. Wiley-IEEE Press.

Buntine, W. (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210.

Burbidge, R., Rowland, J. J., and King, R. D. (2007). Active learning for regression based on query by committee. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 209–218. Springer.

Butenkov, S. (2004). Granular computing in image processing and understanding. In *Proceedings of International Conference on Artificial Intelligence AIA-2004, Innsbruk*, pages 811–816.

Caruana, R. (1998). Multitask learning. In *Learning to Learn*, pages 95–133. Springer.

Caruna, R. (1993). Multitask learning: A knowledge-based source of inductive bias. In *Machine Learning: Proceedings of the Tenth International Conference*, pages 41–48.

Celiberto Jr, L. A., Matsuura, J. P., De Mantaras, R. L., and Bianchi, R. A. (2011). Using cases as heuristics in reinforcement learning: a transfer learning application. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1211.

Chen, M.-Y. and Chen, B.-T. (2015). A hybrid fuzzy time series model based on granular computing for stock price forecasting. *Information Sciences*, 294:227–241.

Chopra, S., Balakrishnan, S., and Gopalan, R. (2013). Dlid: Deep learning for domain adaptation by interpolating between domains. In *ICML Workshop on Challenges in Representation Learning*, volume 2, pages 647–655.

Cireşan, D. C., Meier, U., and Schmidhuber, J. (2012). Transfer learning for latin and chinese characters with deep neural networks. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167. ACM.

Dai, W., Xue, G.-R., Yang, Q., and Yu, Y. (2007a). Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 210–219. ACM.

Dai, W., Xue, G.-R., Yang, Q., and Yu, Y. (2007b). Transferring naive bayes classifiers for text classification. In *AAAI*, volume 7, pages 540–545.

Dai, W., Yang, Q., Xue, G.-R., and Yu, Y. (2007c). Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 193–200. ACM.

Das, S., Abraham, A., and Konar, A. (2008). Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. *Advances of Computational Intelligence in Industrial Systems*, pages 1–38.

Deng, Z., Choi, K.-S., Jiang, Y., and Wang, S. (2014). Generalized hidden-mapping ridge regression, knowledge-leveraged inductive transfer learning for neural networks, fuzzy systems and kernel methods. *IEEE Transactions on Cybernetics*, 44(12):2585–2599.

Duan, L., Xu, D., and Tsang, I. (2012). Learning with augmented features for heterogeneous domain adaptation. *arXiv preprint arXiv:1206.4660*.

Finkel, J. R. and Manning, C. D. (2009). Hierarchical bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 602–610. Association for Computational Linguistics.

Friedman, N. and Koller, D. (2003). Being bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine Learning*, 50(1-2):95–125.

Gönen, M. and Margolin, A. A. (2014). Kernelized bayesian transfer learning. In *AAAI*, pages 1831–1839.

Gong, B., Grauman, K., and Sha, F. (2014). Learning kernels for unsupervised domain adaptation with applications to visual object recognition. *International Journal of Computer Vision*, 109(1-2):3–27.

Guo, Y. and Greiner, R. (2007). Optimistic active-learning using mutual information. In *IJCAI*, volume 7, pages 823–829.

Hadjili, M. L. and Wertz, V. (2002). Takagi-sugeno fuzzy modeling incorporating input variables selection. *IEEE Transactions on Fuzzy Systems*, 10(6):728–742.

Han, J. and Dong, J. (2007). Perspectives of granular computing in software engineering. In *IEEE International Conference on Granular Computing, 2007*, pages 66–66. IEEE.

Hardoon, D. R., Szedmak, S., and Shawe-Taylor, J. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664.

Havens, T. C., Bezdek, J. C., Leckie, C., Hall, L. O., and Palaniswami, M. (2012). Fuzzy c-means algorithms for very large data. *IEEE Transactions on Fuzzy Systems*, 20(6):1130–1146.

Heckerman, D. et al. (1998). A tutorial on learning with bayesian networks. *Nato Asi Series D Behavioural And Social Sciences*, 89:301–354.

Huang, J.-T., Li, J., Yu, D., Deng, L., and Gong, Y. (2013). Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7304–7308. IEEE.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154.

Jiang, J. and Zhai, C. (2007). A two-stage approach to domain adaptation for statistical classifiers. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, pages 401–410. ACM.

Jin, F. and Sun, S. (2008). Neural network multitask learning for traffic flow forecasting. In *IEEE International Joint Conference on Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence)*, pages 1897–1901. IEEE.

Kanamori, T., Hido, S., and Sugiyama, M. (2009). A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10(Jul):1391–1445.

Kandaswamy, C., Silva, L. M., Alexandre, L. A., Santos, J. M., and de Sá, J. M. (2014). Improving deep neural network performance by reusing features trained with transductive transference. In *International Conference on Artificial Neural Networks*, pages 265–272. Springer.

Klenk, M. and Forbus, K. (2009). Analogical model formulation for transfer learning in ap physics. *Artificial Intelligence*, 173(18):1615–1638.

Koçer, B. and Arslan, A. (2010). Genetic transfer learning. *Expert Systems with Applications*, 37(10):6997–7002.

Koivisto, M. and Sood, K. (2004). Exact bayesian structure discovery in bayesian networks. *Journal of Machine Learning Research*, 5(May):549–573.

Kononenko, I. (1993). Inductive and bayesian learning in medical diagnosis. *Applied Artificial Intelligence an International Journal*, 7(4):317–337.

Kulis, B., Saenko, K., and Darrell, T. (2011). What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1785–1792. IEEE.

Lemos, A., Caminhas, W., and Gomide, F. (2011). Multivariable gaussian evolving fuzzy modeling system. *IEEE Transactions on Fuzzy Systems*, 19(1):91–104.

Lewis, D. D. (1992). *Representation and learning in information retrieval*. PhD thesis, University of Massachusetts at Amherst.

Lewis, D. D. and Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Machine Learning Proceedings 1994*, pages 148–156. Elsevier.

Li, W., Duan, L., Xu, D., and Tsang, I. W. (2014). Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1134–1148.

Lim, C.-H., Wan, Y., Ng, B.-P., and See, C.-M. S. (2007). A real-time indoor wifi localization system utilizing smart antennas. *IEEE Transactions on Consumer Electronics*, 53(2).

Lin, T. (1998). Granular computing on binary relations ii: Rough set representations and belief functions. *Rough Sets in Knowledge Discovery*, 1:122–140.

Liu, W., Zhang, H., and Li, J. (2009). Inductive transfer through neural network error and dataset regrouping. In *IEEE International Conference on Intelligent Computing and Intelligent Systems, 2009. ICIS 2009.*, volume 1, pages 777–781. IEEE.

Long, M., Wang, J., Cao, Y., Sun, J., and Philip, S. Y. (2016). Deep learning of transferable representation for scalable domain adaptation. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2027–2040.

Luis, R., Sucar, L. E., and Morales, E. F. (2010). Inductive transfer for learning bayesian networks. *Machine Learning*, 79(1-2):227–255.

Ma, Y., Luo, G., Zeng, X., and Chen, A. (2012). Transfer learning for cross-company software defect prediction. *Information and Software Technology*, 54(3):248–256.

Nguyen, H. V., Ho, H. T., Patel, V. M., and Chellappa, R. (2015). Dash-n: Joint hierarchical domain adaptation and feature learning. *IEEE Transactions on Image Processing*, 24(12):5479–5491.

Niculescu-Mizil, A. and Caruana, R. (2007). Inductive transfer for bayesian network structure learning. In *Artificial Intelligence and Statistics*, pages 339–346.

Oyen, D. and Lane, T. (2013). Bayesian discovery of multiple bayesian networks via transfer learning. In *2013 IEEE 13th International Conference on Data Mining (ICDM)*, pages 577–586. IEEE.

Oyen, D., Lane, T., et al. (2012). Leveraging domain knowledge in multitask bayesian network structure learning. In *AAAI*.

Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Pan, S. J., Zheng, V. W., Yang, Q., and Hu, D. H. (2008). Transfer learning for wifi-based indoor localization. In *Association for the Advancement of Artificial Intelligence (AAAI) Workshop*, page 6.

Pedrycz, W. (2001). *Granular computing: an emerging paradigm*, volume 70. Springer Science & Business Media.

Pedrycz, W. (2013). *Granular computing: Analysis and design of intelligent systems*. CRC Press.

Pedrycz, W. (2014). Allocation of information granularity in optimization and decision-making models: towards building the foundations of granular computing. *European Journal of Operational Research*, 232(1):137–145.

Pedrycz, W., Al-Hmouz, R., Morfeq, A., and Balamash, A. (2013). The design of free structure granular mappings: the use of the principle of justifiable granularity. *IEEE Transactions on Cybernetics*, 43(6):2105–2113.

Pedrycz, W. and Chen, S.-M. (2015). *Granular computing and decision-making: interactive and iterative approaches*, volume 10. Springer.

Pedrycz, W. and Gomide, F. (2007). *Fuzzy systems engineering: toward human-centric computing*. John Wiley & Sons.

Pedrycz, W. and Homenda, W. (2013). Building the fundamentals of granular computing: a principle of justifiable granularity. *Applied Soft Computing*, 13(10):4209–4218.

Pedrycz, W. and Vukovich, G. (2002). Granular computing with shadowed sets. *International Journal of Intelligent Systems*, 17(2):173–197.

Price, K., Storn, R. M., and Lampinen, J. A. (2006). *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media.

Rasmussen, C. E. (2000). The infinite gaussian mixture model. In *Advances in Neural Information Processing Systems*, pages 554–560.

Raykar, V. C., Krishnapuram, B., Bi, J., Dundar, M., and Rao, R. B. (2008). Bayesian multiple instance learning: automatic feature selection and inductive transfer. In *Proceedings of the 25th International Conference on Machine Learning*, pages 808–815. ACM.

Richardson, M. and Domingos, P. (2003). Learning with knowledge from multiple experts. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 624–631.

Roy, D. M. and Kaelbling, L. P. (2007). Efficient bayesian task-level transfer learning. In *IJCAI*, volume 7, pages 2599–2604.

Salakhutdinov, R., Tenenbaum, J., and Torralba, A. (2012). One-shot learning with a hierarchical nonparametric bayesian model. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 195–206.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47.

Settles, B. (2010). Active learning literature survey. 2010. *Computer Sciences Technical Report*.

Settles, B. and Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079. Association for Computational Linguistics.

Settles, B., Craven, M., and Ray, S. (2008). Multiple-instance active learning. In *Advances in Neural Information Processing Systems*, pages 1289–1296.

Shell, J. and Coupland, S. (2012). Towards fuzzy transfer learning for intelligent environments. In *International Joint Conference on Ambient Intelligence*, pages 145–160. Springer.

Shell, J. and Coupland, S. (2015). Fuzzy transfer learning: Methodology and application. *Information Sciences*, 293:59–79.

Shi, X., Liu, Q., Fan, W., and Philip, S. Y. (2013). Transfer across completely different feature spaces via spectral embedding. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):906–918.

Shi, Y. and Eberhart, R. C. (1998). Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*, pages 591–600. Springer.

Shi, Y. and Sha, F. (2012). Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. *arXiv preprint arXiv:1206.6438*.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244.

Silver, D. and Mercer, R. (2001). Selective functional transfer: Inductive bias from related tasks. In *IASTED International Conference on Artificial Intelligence and Soft Computing (ASC2001)*, pages 182–189.

Silver, D. and Mercer, R. (2002). The task rehearsal method of life-long learning: Overcoming impoverished data. *Advances in Artificial Intelligence*, pages 90–101.

Silver, D. and Tu, L. (2008). Image transformation: inductive transfer between multiple tasks having multiple outputs. *Advances in Artificial Intelligence*, pages 296–307.

Silver, D. L. and Mercer, R. E. (2007). Sequential inductive transfer for coronary artery disease diagnosis. In *International Joint Conference on Neural Networks, 2007. IJCNN 2007*, pages 2635–2641. IEEE.

Silver, D. L. and Poirier, R. (2007). Context-sensitive mtl networks for machine lifelong learning. In *FLAIRS Conference*, pages 628–633.

Silver, D. L., Poirier, R., and Currie, D. (2008). Inductive transfer with context-sensitive neural networks. *Machine Learning*, 73(3):313.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Swietojanski, P., Ghoshal, A., and Renals, S. (2012). Unsupervised cross-lingual knowledge transfer in dnn-based lvcsr. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 246–251. IEEE.

Tan, S., Cheng, X., Wang, Y., and Xu, H. (2009). Adapting naive bayes to domain adaptation for sentiment analysis. *Advances in Information Retrieval*, pages 337–349.

Taylor, M. E., Stone, P., and Liu, Y. (2007). Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(Sep):2125–2167.

Thrun, S. (1994). A lifelong learning perspective for mobile robot control. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94.*, volume 1, pages 23–30. IEEE.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, pages 640–646.

Ueki, K., Sugiyama, M., and Ihara, Y. (2010). Perceived age estimation under lighting condition change by covariate shift adaptation. In *2010 20th*

International Conference on Pattern Recognition (ICPR), pages 3400–3403. IEEE.

Wang, C. and Mahadevan, S. (2011). Heterogeneous domain adaptation using manifold alignment. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, pages 1541–1546.

Wilson, A., Fern, A., Ray, S., and Tadepalli, P. (2007). Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1015–1022. ACM.

Wilson, A., Fern, A., and Tadepalli, P. (2012). Transfer learning in sequential decision problems: A hierarchical bayesian approach. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 217–227.

Wood, F. and Teh, Y. W. (2009). A hierarchical nonparametric bayesian approach to statistical language model domain adaptation. In *International Conference on Artificial Intelligence and Statistics*, pages 607–614.

Xu, J., Ramos, S., Vázquez, D., and Lopez, A. M. (2014). Domain adaptation of deformable part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2367–2380.

Xu, Z., Akella, R., and Zhang, Y. (2007). Incorporating diversity and density in active learning for relevance feedback. In *European Conference on Information Retrieval*, pages 246–257. Springer.

Yamauchi, K. (2008). Covariate shift and incremental learning. In *International Conference on Neural Information Processing*, pages 1154–1162. Springer.

Yamauchi, K. (2009). Optimal incremental learning under covariate shift. *Memetic Computing*, 1(4):271.

Yang, P., Tan, Q., and Ding, Y. (2008). Bayesian task-level transfer learning for non-linear regression. In *2008 International Conference on Computer Science and Software Engineering*, volume 1, pages 62–65. IEEE.

Yeh, Y.-R., Huang, C.-H., and Wang, Y.-C. F. (2014). Heterogeneous domain adaptation and classification by exploiting the correlation subspace. *IEEE Transactions on Image Processing*, 23(5):2009–2018.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353.

Zhu, X. (2006). Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3).

Zhu, Y., Chen, Y., Lu, Z., Pan, S. J., Xue, G.-R., Yu, Y., and Yang, Q. (2011). Heterogeneous transfer learning for image classification. In *AAAI*.

Zou, W., Zhu, S., Yu, K., and Ng, A. Y. (2012). Deep learning of invariant features via simulated fixations in video. In *Advances in Neural Information Processing Systems*, pages 3203–3211.

Zuo, H., Zhang, G., Pedrycz, W., Behbood, V., and Lu, J. (2017a). Fuzzy regression transfer learning in takagi–sugeno fuzzy models. *IEEE Transactions on Fuzzy Systems*, 25(6):1795–1807.

Zuo, H., Zhang, G., Pedrycz, W., Behbood, V., and Lu, J. (2017b). Granular fuzzy regression domain adaptation in takagi-sugeno fuzzy models. *IEEE Transactions on Fuzzy Systems*, 26(2):847–858.

Abbreviations

CCA	**c**anonical **c**orrelation **a**nalysis
DE	**d**ifferential **e**volution
EM	**e**xpectation **m**aximization
FCM	**f**uzzy **C**- **m**eans
FHeDA	**f**uzzy **h**eterogeneous **d**omain **a**daptation
FHoDA	**f**uzzy **ho**mogeneous **d**omain **a**daptation
IGMM	**i**nfinite **G**aussian **m**ixture **m**odel
MDP	**M**arkov **d**ecision **p**rocesses
MIL	**m**ultiple **i**nstance **l**earning
MMD	**m**aximum **m**ean **d**iscrepancy
MSE	**m**ean **s**quare **e**rror
NLP	**n**atural **l**anguage **p**rocessing
NN	**n**eural **n**etworks
PSO	**p**artical **s**warm **o**ptimization
RBM	**r**estricted **B**oltzmann **m**achine
RMSE	**r**oot **m**ean **s**quare **e**rror
SDA	**s**tacked **d**enoising **a**utoencoder
SVM	**s**upport **v**ector **m**achine
TS	**T**akagi- **S**ugino